Granularität von Services - Eine ökonomische Analyse

Die Autoren

Dr. Alexander Krammer[1]

Prof. Dr. Bernd Heinrich[2]

Dr. Matthias Henneberger[3]

Dr. Florian Lautenbacher[4]

Autorenadressen

[1]Kernkompetenzzentrum Finanz- und Informationsmanagement
Prof. Dr. Hans Ulrich Buhl
Universität Augsburg
86135 Augsburg
Deutschland
alexander.krammer@wiwi.uni-augsburg.de
http://www.wi-if.de

[2]Institut für Wirtschaftsinformatik, Produktion und Logistik
Universität Innsbruck
6020 Innsbruck
Österreich
bernd.heinrich@uibk.ac.at
http://www.uibk.ac.at/iwi2/

[3]Siemens IT Solutions and Services
Richard-Strauss-Straße 76
81679 München
Deutschland
matthias.henneberger@siemens.com

[4]SysTec-CAx GmbH
Agnes-Pockels-Bogen 1
80992 München
Deutschland
florian.lautenbacher@systec-cax.de
http://www.SysTec-CAx.de

Granularität von Services - Eine ökonomische Analyse

Zusammenfassung
Serviceorientierte Architekturen werden für die Gestaltung von Anwendungs- und Unternehmensarchitekturen intensiv diskutiert. Dennoch wurde die Frage nach einer adäquaten Granularität von Services unter ökonomischen Aspekten bisher kaum erforscht. Je feiner die Granularität von Services zur Realisierung der Funktionen eines Prozesses gewählt wird, desto größer ist die Anzahl der notwendigen Services. Dadurch steigt der Kompositionsaufwand. Sehr grobgranulare Services hingegen besitzen den Nachteil, dass ihr Realisierungsaufwand höher ist und die Chance zur Mehrfachverwendung (bspw. in verschiedenen Prozessen) sinkt. Im Beitrag wird daher ein Modell entwickelt, das die Entscheidung bezüglich der Granularität von Services unter ökonomischen Aspekten unterstützt und auf diese Weise bestehende Freiheitsgrade nutzt, die eine rein fachlich orientierte Festlegung der Granularität noch offenlässt. Das Entscheidungsmodell wird am Fallbeispiel einer Bank demonstriert, um die Anwendbarkeit und den praktischen Nutzen zu veranschaulichen.

Stichworte
Serviceorientierte Architektur; Granularität; Metriken; Ökonomische Bewertung

Granularity of Services - An economic analysis

Abstract
Service-oriented architectures are widely discussed as a design principle for application and enterprise architectures. Nevertheless, an adequate granularity of services from an economical perspective has not yet been researched sufficiently. The finer the granularity to realize the functions of a process, the higher is the number of services and the more effort has to be spent to compose them. In contrast, very course-grained services bear the disadvantages of higher implementation costs and lower reuse potential (e.g. in different processes). The decision model proposed in this paper is to determine an adequate granularity of services from an economical perspective. In this way degrees of freedom, which often exist for the choice of granularity after a domain analysis, can be used. We demonstrate the decision model in the context of a financial services provider in order to illustrate its applicability and practical benefits.

Keywords
Service-oriented architecture; granularity; metrics; value-based software engineering

Vorspann

Die Wahl einer adäquaten Granularität von Services stellt auch ein ökonomisches Problem dar. Eine größere Anzahl an Services zur Realisierung der Funktionen eines Prozesses und damit feingranulare Services versprechen nicht nur einen geringeren Entwicklungs- und Pflegeaufwand sondern auch eine höhere Mehrfachverwendung der Services. Gleichzeitig steigt jedoch der Aufwand für die Komposition der (vielen) Services. Im Beitrag wird diese Granularitätsfragestellung aufgegriffen. Hierzu werden zunächst drei verschiedene Metriken zur Granularitätsmessung und danach ein ökonomisches Modell zur Optimierung der Servicegranularität vorgestellt. Damit werden bestehende Ansätze erweitert, die eine rein fachlich orientierte Festlegung der Granularität fokussieren. Die Analyse eines Praxisbeispiels verdeutlicht, dass ein ökonomisches Optimierungspotenzial bei der Realisierung einer Servicelandschaft vorhanden ist.

1 Problemstellung

Obwohl Serviceorientierte Architekturen (SOA) bereits seit geraumer Zeit diskutiert werden, ist die Frage nach einer adäquaten Granularität von Services unter ökonomischen Aspekten noch nicht zufriedenstellend beantwortet. Zwar wird oftmals gefordert, dass Services grobgranularer als Objekte oder als Komponenten sein sollen und gleichzeitig unter fachlichen Gesichtspunkten zu gestalten sind (Henning 2007; Krafzig et al. 2005; Richter et al. 2005). Jedoch bieten derartige Aussagen einen großen Gestaltungsspielraum. Dies ist insofern kritisch, da die Bedeutung der Granularitätsfrage in vielen Beiträgen ausdrücklich betont (siehe obige Quellen) und gar als „Gretchenfrage“ bezeichnet wird (Melzer et al. 2007, S. 33).

Die Vor- und Nachteile grob- bzw. feingranularer Services lassen sich dabei grundsätzlich wie folgt abwägen (vgl. Aier 2006; Erl 2005, S. 557; Melzer et al. 2007, S. 33): Je feingranularer, desto größer ist die Anzahl der Services bspw. zur Realisierung von Funktionen eines Prozesses und desto aufwändiger wird es, die (Vielzahl der) Services zur Prozessausführung zu komponieren. Grobgranulare Services besitzen dagegen den Nachteil, dass das Potenzial auf Mehrfachverwendung bspw. in verschiedenen Prozessen sinkt (vgl. auch Joachim et al. 2011, S. 450). Dies birgt die Gefahr von Redundanz und einer hohen Variantenanzahl, da mehrere Services u. U. gleiche oder ähnliche Funktionen realisieren. Obwohl Services dann immer noch durch ihre Plattformunabhängigkeit, Verwendung von Standards, lose Kopplung etc. charakterisiert sind (Buhl et al. 2008; Papazoglou 2003; Papazoglou et al. 2006), verschwindet ein entscheidender Vorteil: Wenn im Extremfall Services eine ähnliche (grobe) Granularität aufweisen wie monolithische Anwendungssysteme, geht der (propagierte) Vorteil der mehrfachen Verwendung modularer Softwareartefakte und deren „einfache“ Komposition zu neuen oder geänderten Prozessen verloren.

Bislang wird in der Literatur die Frage nach der Servicegranularität primär aus fachlicher Sicht diskutiert (z. B. Albani et al. 2008; Fiege 2009; Winkler 2007; Winter 2003). Diese Frage stellt jedoch auch ein ökonomisches Problem dar: Wie granular sind Services zu entwickeln, damit der Aufwand für die Entwicklung, Komposition und Pflege der Services minimal ist? Grundlage für eine solche ökonomische Betrachtung ist es, geeignete Metriken zu entwickeln, um Granularität zunächst überhaupt nachvollziehbar messen zu können. Zu beiden Punkten will die Arbeit einen Beitrag liefern. Sie folgt damit der wachsenden Erkenntnis, dass ökonomische Aspekte stärker als bisher für die System- und Servicegestaltung zu berücksichtigen sind (vgl. Value-based Software-Engineering; bspw. Biffl et al. 2005).

Die Arbeit gliedert sich wie folgt: In Abschnitt 2 werden Arbeiten diskutiert, die sich mit der Identifikation und Gestaltung von Services (und Komponenten) auf Basis fachlicher Kriterien befassen. Diese fachlichen Vorgaben bilden dabei den Ausgangspunkt für die weitere ökonomische Betrachtung. In Abschnitt 3 werden für die Messung der Granularität verschiedene

Metriken definiert. Diese sind Voraussetzung, um nachvollziehbare Aussagen wie „Realisierung durch fein- oder grobgranulare Services“ überhaupt treffen zu können. Danach wird ein Modell entwickelt, das die Entscheidung über die Granularität von Services unter ökonomischen Kriterien unterstützt. Dieses Modell bzw. seine Anwendbarkeit wird in Abschnitt 4 am Fallbeispiel einer Bank demonstriert. Abschnitt 5 fasst die wesentlichen Ergebnisse zusammen, würdigt diese kritisch und gibt einen Ausblick auf den zukünftigen Forschungsbedarf.

2 Einordnung in die Literatur

In der Literatur lassen sich eine Reihe von Arbeiten zur Identifikation und Gestaltung von Services insbesondere anhand fachlicher Kriterien finden. Dabei können Services als (Software-)Artefakte einer Systemlandschaft verstanden werden, die Funktionen kapseln (Schelp und Winter 2008, S. 6) und sich durch Eigenschaften wie Modularität und lose Kopplung sowie definierte Schnittstellen auszeichnen (Krafzig et al. 2005, S. 59; zu den Eigenschaften von Services vgl. bspw. Buhl et al. 2008, S. 62; Erl 2005, S. 37,). Häufig wird eine Unterscheidung zwischen technischen und fachlichen Services getroffen, wobei letztere komponierbare Funktionen eines Geschäftsprozesses realisieren und meist als Business oder Enterprise Services bezeichnet werden (vgl. Melzer 2007, S. 32; Schelp und Winter 2008, S. 7). Wir beschränken uns im Folgenden auf Letztere, da sich gerade hier die Frage nach einer Mehrfachverwendung in unterschiedlichen Prozessen - und damit die Frage nach der „richtigen“ Granularität - stellt. Unter Granularität verstehen wir dabei zunächst, in Anlehnung an die Literatur, die Anzahl bzw. den Umfang der durch einen Service realisierten Funktionen (Erl 2007; Galster und Bucherer 2008, S. 400; Thomas et al. 2010, S. 366). Bspw. schreiben Papazoglou und van den Heuvel (2006, S. 423): „Service granularity refers to the scope of functionality exposed by a service“. Nach Boerner und Goeken (2009) beschreibt Granularität ebenso den funktionalen Umfang eines Service. Diese kurze Diskussion des Granularitätsbegriffs wird in Abschnitt 3.2 nochmals mit der Messung der Granularität anhand von Metriken aufgenommen.

Für unsere Untersuchung sind neben Arbeiten, die sich originär mit der Identifikation und Gestaltung von Services befassen, auch Arbeiten zur Identifikation von Komponenten relevant (für einen Vergleich von Ansätzen zur Komponentenidentifikation vgl. Birkmeier und Overhage 2009 sowie zur Serviceidentifikation vgl. Birkmeier et al. 2008). Tabelle 1 stellt ausgewählte Beiträge beider Bereiche dar und greift hierfür in Teilen auf die Klassifikation nach Birkmeier und Overhage (2009) sowie Birkmeier et al. (2008) zurück. Danach werden zunächst die Arbeiten zur Komponentenidentifikation sowie anschließend diejenigen zur Serviceidentifikation diskutiert.

Tabelle 1: *Ausgewählte Ansätze zur Identifikation von Komponenten und Services*

Autoren	Zielsetzung	Vorgehen	Gegenstand	Definition von Granularität	Vorgaben zur Festlegung der Granularität von Komponenten bzw. Services	Fokus	Werkzeugunterstützung	Art der Validierung
Aier 2006	Identifikation von Services als Gruppierung zusammengehöriger Elemente einer Unternehmensarchitektur	Automatisiertes Clustering-Verfahren analysiert Beziehungen zwischen den Elementen einer Ereignisgesteuerten Prozesskette	Services	Anzahl der durch einen Service realisierten Funktionen (implizit)	Granularität abhängig von der Parameterwahl des Clustering-Verfahrens (keine Optimierung)	Fachlich	Ja	Fiktives Fallbeispiel
Albani et al. 2008	Identifikation von Geschäftskomponenten, die fachlich zusammengehören	Verfeinerung von Domänenmodellen zu Komponenten unter Zuhilfenahme von Heuristiken (Top Down)	Komponenten	Keine explizite Definition von Granularität	Präferenzen zu Dekompositions- und Gruppierungsregeln	Fachlich	Ja	Reales Fallbeispiel
Arsanjani et al. 2008	Entwicklung einer Serviceorientierten Architektur unter Berücksichtigung des kompletten Lebenszyklus von Services	Top Down-Analyse der Domäne sowie von Geschäftszielen (Fokus) bei zusätzlicher Bottom Up-Analyse existierender Systeme	Services	Keine explizite Definition von Granularität	Keine Vorgaben	Fachlich	Nein	Darstellung von Fallbeispielen und Beschreibung gesammelter Best Practices
Beverungen et al. 2008	Identifikation von Services auf Basis von Geschäftsprozessen anhand eines Vorgehensmodells	Dekomposition von Geschäftsprozessen unter Berücksichtigung von Sichtbarkeitsaspekten bzgl. Geschäftspartner (Top Down)	Services	Keine explizite Definition von Granularität	Unterscheidung in Prozess- und Basisservices, aber keine detaillierten Vorgaben	Fachlich	Nein	Reales Fallbeispiel
Fiege 2009	Modellierung einer Serviceorientierten Architektur mit Hilfe von Axiomatic Design unter der Wahrung der Architekturziele lose Kopplung, hohe Autonomie und ausgewogene Granularität	Axiomatic Design: Strukturiertes Top Down-Vorgehen mit dem Ausgangspunkt Geschäftsprozesse	Services	Granularität beschreibt den Umfang und die Art der Funktionen. Die funktionale Komplexität wird dabei anhand der Summe der Datenflüsse der Operationen eines Service gemessen.	Dekompositionsregeln und Modellierungsrichtlinien	Fachlich	Nein	Drei reale Fallbeispiele bei Unternehmen
Her et al. 2008	Identifikation von Services ausgehend von Anwendungsfällen	5-stufiges Vorgehen zur Ableitung von Servicespezifikationen aus Anwendungsfällen und Geschäftsprozessen	Services	Keine explizite Definition von Granularität	Granularität wird bereits für Anwendungsfälle bestimmt. Sind diese wiederum in andere Anwendungsfälle inkludiert, wird ein Subprozess und daraufhin ein so genannter Composite-Service identifiziert.	Fachlich	Nein	Fallbeispiel
Kim und Chang 2004	Identifikation von Komponenten unter Berücksichtigung von Kohäsion und Kopplung	Clustering-Verfahren auf Basis von Anwendungsfalldiagrammen (Bottom Up)	Komponenten	Anzahl der realisierten Anwendungsfälle (implizit)	Granularität abhängig von der Parameterwahl des Clustering-Verfahrens (keine Optimierung)	Fachlich, z. T. Technisch	Nein	Keine (nur Diskussion und Bewertung im Vergleich zu anderen Ansätzen)

Lee et al. 2001	Identifikation von Komponenten unter Berücksichtigung von Kohäsion und Kopplung	Clustering-Verfahren auf der Basis von Klassendiagrammen (Bottom Up)	Komponenten	Anzahl der Klassen (implizit)	Granularität abhängig von der Kohäsion und Kopplung der Klassen und der Parameterwahl des Clustering-Verfahrens	Fachlich / Technisch	Nein, nur Clustering-Algorithmus	Reales Fallbeispiel
Offermann 2008	Konzeption betrieblicher Software mit einer Service-orientierten Architektur	Top Down-Analyse des Unternehmens konsolidiert mit Bottom Up-Analyse von Bestandssystemen	Services	Keine explizite Definition von Granularität	Keine Vorgaben	Fachlich	Ja (aber nur zur Servicemodellierung, nicht zur Granularitätsoptimierung)	Reales Fallbeispiel und Laborexperiment
Wang et al. 2006	Identifikation von Komponenten unter besonderer Berücksichtigung der Stabilität des Geschäftsmodells	Verfahren aufbauend auf dem sogenannten „Feature Tree“, der die Features (einer Domäne) und deren Abhängigkeit darstellt und nach deren Stabilität differenziert	Komponenten	Anzahl bereitgestellter Features / Funktionen oder implementierter Entitäten	Optimierung der Granularität auf Basis des „Feature Tree“	Fachlich und Ökonomisch (siehe auch Wang et al. 2005)	Als "Future Work" deklariert	Qualitativer Vergleich mit anderen Identifikationsansätzen und Veranschaulichung dieses Vergleichs an einem kleinen Beispiel
Winkler 2007	Identifikation von Services unter Berücksichtigung von Mehrfachverwendung	Funktionale Analyse durch schrittweise Zerlegung von Aktionen in Aktivitätsdiagrammen (Top Down)	Services	Orientiert an den Ebenen der Zerlegung (implizit)	Dekompositionsregeln	Fachlich	Nein	Reales Fallbeispiel
Winter 2003; Schelp und Winter 2008	Identifikation von Services, die fachlich zusammengehören	Analyse der Beziehungen zwischen Datenobjekten und Funktionen und deren Gruppierung anhand einer Matrix, mehrdimensional	Services	Funktionsumfang eines Service (implizit)	Dekompositions- und Gruppierungsregeln	Fachlich	Nein	Reale Fallbeispiele mit vier Unternehmen

Arbeiten zur Komponentenidentifikation

Albani et al. (2008) identifizieren in ihrem Ansatz *Business Component Identification* (Geschäfts-)Komponenten auf der Basis von Informationsobjekten und Prozessmodellen sowie den Beziehungen zwischen beiden. Mit Hilfe eines Clustering-Verfahrens wird eine Partitionierung von Komponenten automatisiert vorgenommen, wobei Präferenzen geäußert werden können. Diese sind im Ansatz ebenso berücksichtigt, wie die Art und Häufigkeit der Beziehungen zwischen Informationsobjekten und Aktionen des Prozessmodells. Dabei werden ebenfalls die Ziele einer hohen Kohäsion und niedrigen Kopplung bei der Komponentenidentifikation berücksichtigt.

Daneben wird im Komponentenbereich auch von anderen Clustering-Ansätzen die Granularitätsfragestellung als explizites Entwurfsziel verfolgt (vgl. Kim und Chang 2004; Lee et al. 2001). Diese basieren auf mathematisch-statistischen Verfahren, welche die Kohäsion bzw. Kopplung einzelner Elemente eines Systems - oft Klassen im Sinne der Objektorientierung - bspw. anhand der Anzahl wechselseitiger Beziehungen messen und daraus lose gekoppelte Cluster ableiten, die jeweils eine hohe Kohäsion besitzen. Diese Cluster bilden anschließend den Vorschlag für die Komponenten. Demnach werden je nach Parameterwahl für das Clustering-Verfahren Komponenten unterschiedlicher Granularität identifiziert. Auf die konkrete Wahl der Parameter, insbesondere unter ökonomischen Aspekten, wird in den genannten Beiträgen nicht eingegangen.

Wang et al. (2006) entwickeln ihren Ansatz STCIM (Stability Based Component Identification Method) mit der Zielsetzung, Komponenten primär unter Berücksichtigung der unterschiedlichen Stabilität (von Teilen) des Geschäftsmodells zu identifizieren. Stabilität als Größe wird dabei als der Umfang und die Anzahl von geschäftsbezogenen Änderungen definiert, d. h. je weniger Änderungen vorliegen, desto stabiler sind die Teile des Geschäftsmodells und desto weniger müssen die zugeordneten Komponenten angepasst werden. Stabilität wird demnach als Indikator für die Gestaltung grobgranularer Komponenten gesehen und vice versa (bspw. Wang et al. 2006, S. 2). Um diese Zielsetzung zu erreichen, definieren Wang et al. (2006) zunächst eine Baumstruktur (so genannter „Feature Tree“), die das Ergebnis einer fachlichen Analyse der Domäne widerspiegelt. Diese Baumstruktur enthält neben den Features (im Sinne von Funktionen) auch deren Beziehungen. D. h. Features werden schrittweise verfeinert, um die verschiedenen Ebenen des Baumes zu definieren. In Abhängigkeit davon, auf welcher Ebene des Baumes eine Komponente realisiert wird, ergeben sich Komponenten unterschiedlicher Granularität.

Daneben sind speziell die von Wang et al. (2006, S. 6) explizit diskutierten ökonomischen Größen Kompositions- bzw. Änderungsaufwand interessant. So wird angeführt, dass je grobgranularer Komponenten sind, desto geringer ist deren Kompositionsaufwand und vice

versa. Dagegen steigt der Änderungsaufwand, je grobgranularer Komponenten sind und vice versa. Beides wird jedoch ausschließlich argumentativ festgestellt, da der exakte funktionale Zusammenhang zur Berechnung, bspw. des Kompositionsaufwands, nicht definiert oder ausgeführt wird. Interessanterweise werden auch die Aufwandsgrößen im eigenen Ansatz STCIM nicht weiter betrachtet.

Darüber hinaus wird in Wang et al. (2005, S. 231) ein Optimierungskalkül für die Identifikation und Gestaltung von Komponenten vorgestellt. Hier werden verschiedene Zielgrößen für eine Komponente *c* definiert, wie bspw. Reusability (Variable *R(c)*), Reuse Cost (*RC(c)*) oder Reuse efficiency (*RE(c)*) und deren jeweilige Zielvorschrift (Minimierung oder Maximierung). Diese Zielgrößen werden danach in einer einzigen Zielfunktion integriert, welche über die Menge aller Komponenten *c* zu maximieren ist. Jedoch wird leider auch in (Wang et al. 2005, S. 231) - analog zu oben - für die einzelnen Zielgrößen kein funktionaler Zusammenhang zu deren Berechnung definiert. Insofern ist nicht nachvollziehbar, welche Eigenschaften und Bestandteile das ökonomische Kalkül besitzt bzw. wie es für die Komponentenidentifikation genau zu nutzen ist. Daneben ist ebenso anzumerken, dass die Zielgrößen über unterschiedliche Maßeinheiten verfügen. Bspw. dürfte die Zielgröße Reuse Cost (*RC(c)*) eine Geldeinheit als Maßeinheit besitzen, wohingegen Größen wie *Cohesion(c)* oder *Coupling(c)* als „semantische Nähe“ (ohne Maßeinheitsangabe) definiert sind. Inwiefern die Verknüpfung der einzelnen Zielgrößen in der dargestellten Zielfunktion zu einem interpretierbaren Gesamtergebnis führt wird daher nicht klar.

Arbeiten zur Serviceidentifikation

Im Gegensatz zu einzelnen Ansätzen der Komponentenidentifikation zeigt die Tabelle 1, dass die Serviceidentifikation bisher fast ausschließlich fachlich diskutiert wird, wobei Regeln zur Festlegung der Granularität nur zum Teil explizit angegeben werden.

Aier (2006) schlägt bspw. einen Clustering-Algorithmus zur fachlich orientierten Modularisierung einer Anwendungssystemlandschaft vor. Damit soll u. a. auch die Servicegranularität bestimmt werden. Winter (2003) lehnt sich an den Business Systems Planning-Ansatz von IBM (1984) an und schlägt drei Dimensionen zur Strukturierung von Anwendungssystemlandschaften vor (Funktion, Informationsobjekt, Leistung/Organisation), was in Nachfolgearbeiten (vgl. Schelp und Winter 2008) auch auf die Serviceidentifikation übertragen wird. Im Rahmen des SOMA-Ansatzes (Service-Oriented Modeling Architecture) von IBM (Arsanjani et al. 2008) wie auch bei Offermann (2008) erfolgt die Identifikation und Gestaltung von Services sowohl Bottom Up, ausgehend von bestehenden Anwendungen oder Komponenten, als auch Top Down, z. B. ausgehend von Prozessmodellen oder Geschäftszielen. Winkler (2007) zerlegt hingegen Aktivitätsdiagramme in mehreren iterativen Schritten in atomare Basisaktionen oder -funktionen, um zu entscheiden, welche davon einzeln oder integriert in

einem Service realisiert werden sollen. Damit ähnelt dieser Ansatz dem Vorgehen von Her et al. (2008), die ebenfalls aus Prozessmodellen und Anwendungsfällen durch iterative Verfeinerung Services identifizieren. Beverungen et al. (2008) berücksichtigen zusätzlich, ob ein Prozessschritt auch für Geschäftspartner sichtbar sein soll, um Services zu identifizieren. Fiege (2009) überträgt das sogenannte Axiomatic Design, einem aus der industriellen Produktion stammenden Verfahren, auf die Serviceidentifikation. Im Sinne eines Top Down-Verfahrens werden mögliche Lösungen zur Erfüllung der vorab festgelegten Anforderungen durch Zuordnung und Dekomposition in Form von Entwurfsmatrizen abgebildet. Die dort dargestellten Beziehungen zwischen funktionalen Anforderungen und Designparametern sollen die Identifikation und Gestaltung von Services unter der Prämisse loser Kopplung, hoher Autonomie und „ausgewogener" Granularität ermöglichen.

Bei vielen der oben diskutierten Ansätze bestehen nach ihrer Anwendung i. d. R. noch Freiheitsgrade für die Serviceidentifikation. Je nachdem wie stark bspw. Aktivitätsdiagramme verfeinert oder Funktionen zerlegt werden, können Services unterschiedlicher Granularität entstehen. Hier knüpft die vorliegende Arbeit mit der Frage an, wie diese noch vorhandenen Freiheitsgrade bei der Servicegranularität unter ökonomischen Aspekten zu nutzen sind. Insofern werden die obigen, fachlich orientierten Arbeiten erweitert. Dies entspricht einem zweistufigen Vorgehen:

1. Die Anwendung eines fachlich orientierten Ansatzes zur Serviceidentifikation führt zu jeweils alternativen Servicekandidaten (im Sinne von Freiheitsgraden), die sich im Umfang der realisierten Funktionen unterscheiden.
2. Basierend auf der Ermittlung von Entwicklungs- und Pflegeaufwand und unter Berücksichtigung des Potenzials auf Mehrfachverwendung wird eine ökonomische Optimierung durchgeführt, welche die vorhandenen Freiheitsgrade der fachlichen Serviceidentifikation nutzt.

Teilweise sehen Ansätze zur Modellierung von SOA bereits explizit einen Schritt zur Konsolidierung möglicher Servicekandidaten vor (bspw. Offermann 2008, S. 467), der um eine ökonomische Bewertung erweitert werden kann. Eine derartige ökonomische Betrachtung bietet jedoch bisher keiner der untersuchten Ansätze. Dies gilt auch für den Komponentenbereich. Obwohl insbesondere der Arbeit von Wang et al. (2005) bereits einzelne Aufwandsgrößen und ein Kalkül zugrunde liegen, sind diese Ausführungen zur Beantwortung der Frage der Servicegranularität nur begrenzt geeignet, da ökonomische Zielgrößen und Zusammenhänge (wie hängt bspw. der Aufwand zur Servicerealisierung vom Funktionsumfang ab?) nicht behandelt oder konkretisiert werden. Im Folgenden wird daher ein Ansatz vorgestellt, der dies berücksichtigt.

3 Granularität von Services - Eine ökonomische Analyse

Die Definition des Funktionalitäts- und Servicegraphs in Abschnitt 3.1 bildet den Ausgangspunkt für die danach vorgestellten Granularitätsmetriken und die ökonomische Betrachtung im Abschnitt 3.3.

3.1 Funktionalitäts- und Servicegraph (FSG)

Einige der obigen Ansätze schlagen zur Serviceidentifikation eine Aggregation oder Disaggregation von Funktionalitäten unter fachlichen Aspekten vor (im Weiteren wird der Begriff der Funktionalität äquivalent zu dem der Funktion verwendet). Dabei stellt sich die Frage, wie das Ergebnis dieser Analyse für unsere Zwecke geeignet abzubilden ist. Im Folgenden wird ein Graph - der FSG - zur Repräsentation der Ergebnisse definiert. Analog zu einigen fachlichen Ansätzen soll der FSG die Disaggregationsbeziehungen zwischen Funktionalitäten repräsentieren. Der Betrachtungsgegenstand Service bedarf jedoch - im Vergleich bspw. zu Wang et al. (2006) - u. a. folgender Erweiterung:

1. Um eine Mehrfachverwendung von Funktionalitäten darstellen zu können, verwenden wir einen gerichteten, azyklischen Graphen anstatt eines Baumes. Bei Wang et al. (2006, S. 4) ist der „Feature Tree“ dagegen analog zu den Eigenschaften eines Baumes dadurch gekennzeichnet, dass eine Sohnfunktionalität einschränkend nur eine Elternfunktionalität besitzen darf.
2. Services können zur Durchführung unterschiedlicher Prozesse komponiert werden. Dazu muss der Graph mehrere Quellknoten (Funktionalitäten ohne eingehende Kanten) enthalten können, die als Prozesse zu interpretieren sind. Bei Wang et al. (2006, S. 4) besitzt der „Feature Tree“ einschränkend nur eine Wurzel, d. h. mehrere Prozesse sind nicht als Wurzelknoten darstellbar.

Die Annahmen und Definitionen für den FSG lauten demnach wie folgt:

(A1) Der FSG ist ein gerichteter, azyklischer Graph $G=(N, E)$ mit den Funktionalitäten $m \in M$ mit $M \subseteq N$ als Knoten sowie den Disaggregationsbeziehungen $(m_i, m_j) \in E$ zwischen den Funktionalitäten m_i und m_j als gerichtete Kanten. Die Disaggregation einer Funktionalität m_i in die Funktionalitäten m_j, …, m_{j+n} erfolgt disjunkt und vollständig.

Eine gerichtete Kante (m_i, m_j) bedeutet „Funktionalität m_j ist Teil der Funktionalität m_i“. Im FSG sind alle Funktionalitäten und Disaggregationsbeziehungen der betrachteten fachlichen Domäne abgebildet.

Jeder Quellknoten des FSG, d. h. ein Knoten ohne eingehende Kanten, wird als *Prozess* bezeichnet (mit P als Menge aller Prozesse, $P \subseteq M$). Jede Senke des FSG, d. h. ein Knoten ohne ausgehende Kanten, wird als *Basisfunktionalität* bezeichnet (mit B als Menge aller Basisfunktionalitäten, $B \subseteq M$). Die inneren Knoten des Graphen, also Funktionalitäten, die we-

der Prozesse noch Basisfunktionalitäten sind, werden *vorausgehende Funktionalitäten* genannt. $V \subseteq M$ ist die Menge aller vorausgehenden Funktionalitäten.

Zudem wird eine offene Folge von Knoten und Kanten $m_0, (m_0, m_1),\ldots, (m_{n-1}, m_n), m_n$ als *Weg* $w(m_0, m_n)$ bezeichnet, mit dem Anfangsknoten m_0 und dem Endknoten m_n. Ist der Anfangsknoten eines Weges ein Prozess (d. h. $m_0 \in P$) und der Endknoten eine Basisfunktionalität (*d. h.* $m_n \in B$), so entspricht dies einem *Gesamtweg*. Die Länge $d(m_0, m_n)$ eines Weges $w(m_0, m_n)$ ist im azyklischen FSG definiert als die Anzahl der zugehörigen Kanten des Weges *w.*

Die Disaggregationsbeziehungen zwischen Funktionalitäten und damit die Wege lassen sich anhand einer Adjazenzmatrix I_{MM}: $M \times M$ darstellen, wobei $I_{m_i,m_j} = 1$ genau dann gilt, wenn $(m_i, m_j) \in E$ existiert. Ansonsten gilt $I_{m_i,m_j} = 0$.

Im FSG soll auch das Ergebnis des späteren ökonomischen Entscheidungsmodells dargestellt werden, d. h. welche Funktionalitäten konkret mit einem Service zu realisieren sind:

(A2) Jeder Service $s_i \in S$ mit $S \subseteq N$ wird genau einer Funktionalität $m_j \in V \cup B$ durch eine gerichtete Kante $(s_i, m_j) \in E$ zugeordnet. Ein Service s_i realisiert die Funktionalität m_j vollständig, d. h. auch alle Funktionalitäten m_k, für die es einen Weg $w(m_j, m_k)$ gibt und bietet genau diese Funktionalität über seine Schnittstelle an.

Eine Funktionalität m_j für die gilt $\exists (s_i, m_j) \in E$ heißt *realisierte Funktionalität*. Zur Mehrfachrealisierung einer Funktionalität kommt es dann, wenn eine Funktionalität direkt oder über vorausgehende Funktionalitäten durch verschiedene Services realisiert wird. Die Matrix I_{SM}: $S \times M$ repräsentiert die Servicezuordnung, wobei $I_{s_i,m_j} = 1$ genau dann gilt, wenn $(s_i, m_j) \in E$ mit $s_i \in S$ und $m_j \in V \cup B$. Ansonsten gilt $I_{s_i,m_j} = 0$.

Die obigen Ausführungen sind in Abbildung 1 anhand eines einfachen Beispiels veranschaulicht. Als Ergebnis einer fachlichen Analyse liegt hier die Disaggregation zweier Prozesse m_0 und m_{10} in die Funktionalitäten m_1 und m_4 bzw. m_{11} vor usw. Die Funktionalität m_9 wird dabei in beiden Prozessen benötigt. Eine *mögliche* Realisierung der Funktionalitäten durch die Services s_1 bis s_5 ist ebenfalls dargestellt. Der Service s_3 realisiert bspw. die Funktionalität m_6 direkt und damit auch die Basisfunktionalitäten m_8 und m_9. Da Service s_5 über die Funktionalität m_{11} ebenfalls m_9 realisiert, wird m_9 im Beispiel zweifach umgesetzt.

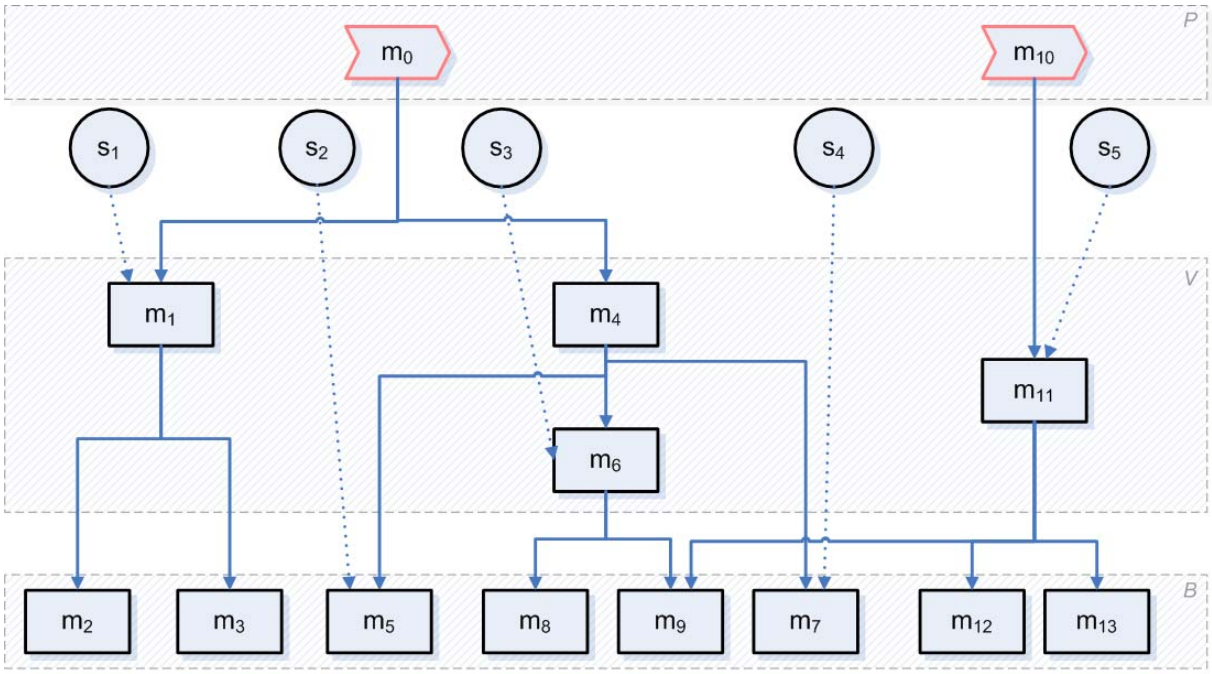


***Abbildung 1**: Funktionalitäts- und Servicegraph (FSG) im Beispiel*

3.2 Granularitätsmetriken

Bestehende Ansätze treffen zwar Aussagen zur Granularität - wie bspw. grob- vs. feingranulare Services - und z. T. ergibt sich das jeweilige Begriffsverständnis implizit aus dem Kontext, nur wenige definieren jedoch diesen Begriff explizit oder formal (vgl. Tabelle 1). Im Folgenden stellen wir daher drei Granularitätsmetriken vor, die unterschiedliche Sichtweisen operationalisieren und diskutieren jeweilige Vor- und Nachteile. Mit Hilfe derartiger Metriken soll die Granularität verschiedener Services gemessen und damit verglichen werden können. Für die Metrikberechnung kann dabei auf den FSG zurückgegriffen werden. Ausgangspunkt ist ein Service s_i, der eine Funktionalität m_j realisiert und dessen Granularität zu bestimmen ist.

Tiefenmaß

Zuerst wird ein Tiefenmaß vorgestellt. Es gibt grundsätzlich an, an welcher Position sich eine realisierte Funktionalität (Service) auf einem (Gesamt)Weg im FSG befindet. Eine einfache Angabe der Weglänge vom Prozess zur realisierten Funktionalität reicht jedoch nicht aus. Dies hat zwei Gründe: Erstens ist dieser Wert wenig aussagekräftig, wenn die Längen der Gesamtwege im FSG stark unterschiedlich sind. So könnte für zwei Services die gleiche Granularität berechnet werden, obwohl der eine Service eine Basisfunktionalität realisiert, während der andere Service eine vorausgehende Funktionalität mit vielen nachfolgenden Funktionalitäten realisiert. Zweitens ist die Angabe einer Weglänge bei einem Graph - anstatt eines Baums - nicht eindeutig, wenn eine Funktionalität in mehreren Wegen enthalten ist. Beides motiviert die Entwicklung folgender Metrik als Operationalisierung des Tiefenmaßes:

Grundlage der Metrik ist die Weglänge vom Prozess bis zur realisierten Funktionalität im Verhältnis zur Gesamtweglänge. Für eine realisierte Funktionalität m_j, die Teil eines Gesamtweges $w(m_p, m_n)$ ist, wird so der Wert $z_w = \frac{d(m_p, m_j) - 1}{\max[1, d(m_p, m_n) - 1]}$ berechnet. Im Zähler als auch im Nenner ist eins abzuziehen, da die Weglänge jeweils ausgehend vom Prozess $m_p \in P$ berechnet wird. Der Wertebereich von z_w ist auf das Intervall *[0; 1]* normiert. So führt die Realisierung einer Basisfunktionalität (maximal feingranular) zum Wert eins, während der Wert null bei einer Realisierung der direkt dem Prozess m_p nachfolgenden Funktionalität resultiert (maximal grobgranular). Der Wert z_w wird für alle Wege berechnet, in denen die realisierte Funktionalität m_j enthalten ist. Über diese Werte wird anschließend das arithmetische Mittel gebildet. Sei *W* die Menge aller Wege *w,* die m_j enthalten, so gilt für die Tiefenmetrik:

$$g_T(s_i) = \frac{\sum_{w \in W} z_w}{|W|}$$

Die Metrik behält dabei die Normierung auf den Wertebereich *[0; 1]*.

Breiten- und Tiefenmaß

Das Tiefenmaß besitzt u. a. dann Nachteile, falls eine realisierte Funktionalität viele direkte und indirekt nachfolgende Funktionalitäten besitzt. Hier liefert das Breiten- und Tiefenmaß aussagekräftige Werte. Ein Service ist demnach umso grobgranularer, je mehr Funktionalitäten er insgesamt (direkt und indirekt) realisiert. Maximal feingranular ist der Service hingegen wieder, wenn er eine Basisfunktionalität realisiert.

Grundlage dieser Metrik ist die Anzahl n_{m_j} der durch einen Service direkt oder indirekt realisierten Funktionalitäten. Dieser Wert wird durch n_{m_a} geteilt, mit m_a als direkt dem Prozess nachfolgende Funktionalität, und von eins subtrahiert: $z_w = 1 - \frac{n_{m_j} - 1}{\max(1; n_{m_a} - 1)}$. Analog zum Tiefenmaß ist der Wertebereich auf *[0; 1]* normiert. Wird eine Basisfunktionalität direkt durch einen Service realisiert, ergibt sich ein Wert von eins (wegen $n_{m_j} - 1 = 0$). Im Falle der Realisierung einer direkt dem Prozess nachfolgenden Funktionalität entsprechen sich die Werte ($n_{m_j} = n_{m_a}$). Daraus resultiert z_w gleich null (außer bei einer Basisfunktionalität). Die Normierung auf *[0; 1]* ist zugleich auch ein Vorteil gegenüber den Metriken von Wang et al. (2006, S. 5-6), bezogen auf Features und die dort definierte Baumstruktur, sowie gegenüber Haesen et al. (2008) und deren „Default Functionality Granularity“, da hier jeweils eine Metrik ohne Normierung vorgeschlagen wird. Diese fehlende Normierung schränkt den Vergleich der Metrikwerte verschiedener Services ein.

Falls eine Funktionalität m_j wiederum Bestandteil mehrerer Wege ausgehend von *verschiedenen* Quellknoten m_p oder vorangehenden Funktionalitäten m_a ist, kann - analog zu oben - das arithmetische Mittel g_{BT} über die Werte dieser Wege gebildet werden.

Beide bisherigen Maße sind zwar - bspw. im Vergleich zur Messung bei Wang et al. (2006) - bereits aussagekräftiger. Sie stützen sich jedoch rein auf die Anzahl der Funktionalitäten. D. h., auch wenn zwei Services nach den beiden obigen Maßen eine identische Granularität aufweisen, können sie sich dennoch anhand des zu realisierenden Funktionalitätsumfangs und bspw. des damit verbundenen Entwicklungsaufwands wesentlich unterscheiden. Letzteres wird nicht transparent, was zu einem dritten Maß führt.

Größenmaß

Die Frage, wie der Umfang von Funktionalitäten gemessen bzw. ex ante vor Realisierung geschätzt werden kann, wird in der Literatur bspw. im Bereich der Aufwandsschätzung untersucht. Hier werden Maßeinheiten wie Lines of Code (LOC) oder Function Points genutzt. LOC werden im COCOMO-Ansatz, einem Verfahren zur Aufwandsschätzung von Softwareentwicklungsprojekten, verwendet und geben an, wie viele Zeilen Quellcode für ein Programm ggf. geschrieben werden müssen (vgl. Boehm et al. 2000, Wehrmann und Gull 2006). In COCOMO gehen zur Berechnung der erforderlichen Personenmonate die LOC der einzelnen zu implementierenden Softwarebestandteile ein, die mit Aufwandsmultiplikatoren, Skalenfaktoren sowie einzelnen Kalibrierungsfaktoren verrechnet werden. Zur Ermittlung der LOC können entweder historische Daten aus anderen Projekten, Expertenschätzungen oder algorithmische Verfahren Verwendung finden. Aufwandsfaktoren werden danach multiplikativ und Skalenfaktoren exponentiell mit den zugehörigen LOC verrechnet. Im später dargestellten Fallbeispiel wurden LOC gewählt, wobei auch andere Maßeinheiten sowohl in der Metrik als auch im Entscheidungsmodell verwendbar sind.

Voraussetzung für das Größenmaß ist die Angabe des Umfangs $size_{m_j}^{func}$ (bspw. in LOC) für eine Basisfunktionalität m_j. Der Umfang vorausgehender Funktionalitäten ergibt sich dann jeweils aus dem Umfang der nachfolgenden Funktionalitäten (disjunkte und vollständige Disaggregation gemäß (A1)) zuzüglich des Umfangs der Kompositionslogik $size_{m_j}^{comp}$ (vgl. dazu auch Erl 2007). Letztere enthält u. a. Information für die Steuerung, die Integration und den Aufruf nachfolgender Funktionalitäten. Der Umfang $size_{m_j}$ einer Funktionalität m_j ergibt sich damit zu:

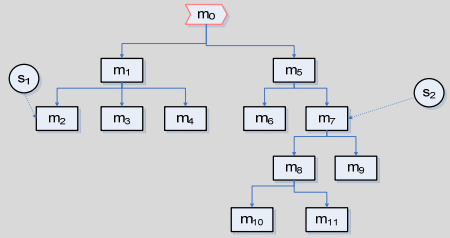
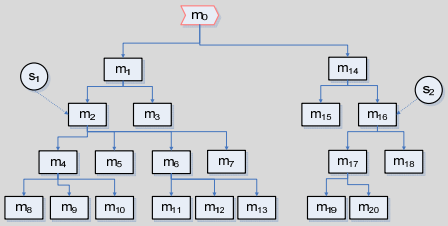
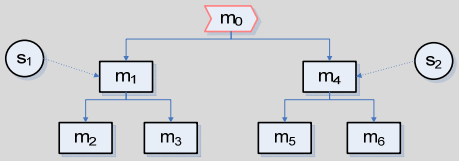
$$size_{m_j} = \begin{cases} size_{m_j}^{func} & \text{falls } m_j \in B \\ size_{m_j}^{comp} + \sum_{i=1}^{|M|} I_{m_j,m_i} \cdot size_{m_i} & \text{falls } m_j \in V \end{cases}$$

Auch hier ist wiederum eine Normierung möglich. Für eine realisierte Funktionalität m_j, die Teil eines Gesamtweges $w(m_p, m_n)$ mit m_p, $(m_p, m_a), \ldots, (m_{n-1}, m_n)$, m_n ist, resultiert

$z_w = 1 - \left[\frac{size_{m_j}}{size_{m_a}} \right]$. Gilt $m_j = m_a$, d. h. eine direkt dem Prozess m_p nachfolgende Funktionalität m_a wird realisiert, dann ist z_W für diese maximal grobgranulare Servicerealisierung null. Bei Realisierung einer Basisfunktionalität wird der Wert z_W größer. Man erhält einen Wertebereich für z_W von *[0; 1[.* Ist eine Funktionalität m_j wiederum Bestandteil mehrerer Wege, kann - analog zu oben - das arithmetische Mittel g_G über die Werte dieser Wege gebildet werden.

In Ergänzung zu den beiden ersten Maßen können die Aussagen des Größenmaßes vor allem dann einen Mehrwert liefern, wenn sich die Umfänge der einzelnen Funktionalitäten stark unterscheiden. Tabelle 2 fasst die Metriken zusammen:

***Tabelle 2** Granularitätsmetriken*

Maß	Beschreibung und Definition	Eignung / Einschränkungen für den Einsatz	Verdeutlichung der Idee der konkreten Metrik (Beispiele)
Tiefenmaß	Metrik misst die Granularität eines Service (realisierte Funktionalität) anhand seiner Position im FSG: Weglänge vom Prozess bis zur realisierten Funktionalität im Verhältnis zur Gesamtweglänge	1. Die Metrik bezieht sich auf die Wege im FSG und führt dann zu gut interpretierbaren Ergebnissen, falls zwei oder mehrere realisierte Funktionalitäten die (annähernd) gleiche Anzahl direkt und indirekt nachfolgender Funktionalitäten besitzen. 2. Sind Funktionalitäten hinsichtlich ihres Umfangs (bspw. LOC) stark unterschiedlich, so ist die Metrik wenig aussagekräftig.	 Aufgrund verschieden langer Gesamtwege ist der Service s_2 mit $g_T(s_2)=0,39$ nach der Metrik für das Tiefenmaß grobgranularer als der Service s_1 mit $g_T(s_1)=1$. Ein bloßer Vergleich der Anzahl vorausgehender Funktionalitäten würde hier die gleiche Granularität ausweisen.
Breiten- und Tiefenmaß	Metrik misst die Granularität eines Service anhand der Anzahl direkt und indirekt nachfolgender Funktionalitäten	1. Metrik führt zu gut interpretierbaren Ergebnissen, falls Servicerealisierungen verglichen werden, die sich in der Anzahl direkter und indirekter Nachfolgefunktionalitäten unterscheiden. 2. Sind Funktionalitäten hinsichtlich ihres Umfangs (bspw. LOC) stark unterschiedlich, so ist die Metrik wiederum wenig aussagekräftig.	 Hier sind die beiden Services nach der Tiefenmetrik (g_T) noch nahezu gleich granular. Jedoch zeigen sich nach der Metrik für das Breiten- und Tiefenmaß nachvollziehbare Unterschiede mit $g_{BT}(s_1)=1/6$ vs. $g_{BT}(s_2)=1/3$. D. h. s_1 ist im Vergleich zu s_2 wesentlich grobgranularer.
Größenmaß	Metrik misst die Granularität eines Service anhand seines Umfangs (bspw. gemessen in LOC)	1. Metrik weist die Unterschiede im Umfang einzelner Funktionalitäten im FSG aus. 2. Metrik betrachtet nicht die Disaggregationsbeziehungen im FSG. 3. Metrik ist insbesondere dann geeignet, wenn die Granularität als Indikator für den Realisierungsaufwand - hier ausgedrückt in LOC - eines Service genutzt wird.	 Die Services s_1 und s_2 sind zwar nach den beiden oberen Metriken gleichgranular. Besitzen die realisierten Funktionalitäten m_1 bzw. m_4 jedoch einen unterschiedlichen Umfang, so wird dies erst mit der Metrik für das Größenmaß transparent.

Die vorgestellten Maße systematisieren verschiedene (oftmals implizite) Verständnisse des Begriffs Granularität in der Literatur. So werden bspw. durch Winkler (2007) im Rahmen ihrer

funktionalen Zerlegung ähnlich einem Tiefenmaß die Dekompositionsbeziehungen ausgewertet. Das dem Breiten- und Tiefenmaß zugrunde liegende Verständnis findet sich ähnlich bspw. bei Aier (2006) und Fiege (2009), wobei letztere explizit neben der Anzahl an Funktionen (Breiten- und Tiefenmaß) auch den Abstraktionsgrad der Services (ähnlich zum Tiefenmaß) diskutieren. Clustering-Verfahren in der Komponentenorientierung, die Elemente wie bspw. Klassen zu Komponenten aggregieren, legen eine implementierungsnahe Granularitätsdefinition ähnlich dem Größenmaß nahe, auch wenn dies i. d. R. nicht expliziert wird.
Anhand obiger Maße lässt sich aber nicht nur die Granularität einzelner Services messen. Aggregiert man vielmehr die Granularität der Services im gesamten FSG, so lässt sich ein Metrikwert für ganze Servicelandschaften angeben (vgl. dazu auch das in Abschnitt 4 vorgestellte Softwarewerkzeug). Insofern können auch verschiedene Gesamtlösungen des ökonomischen Entscheidungsmodells hinsichtlich der Servicegranularität analysiert bzw. verglichen werden. D. h. die Metrikwerte dienen im Weiteren *nicht* als Modellinput. Vielmehr soll der resultierende Modelloutput hinsichtlich seiner Granularität bewertet werden (ist z. B. eine Lösung fein- oder grobgranular?).

3.3 Ökonomisches Entscheidungsmodell

Um zur Forschungsfrage beizutragen, wie granular Services unter ökonomischen Aspekten definiert werden sollen, gilt es die dafür relevanten Aufwandsgrößen zu identifizieren. Wir beschränken uns dabei bewusst auf die Aufwandsseite, da nach erfolgter fachlicher Analyse die Funktionalitäten (die zu Erträgen führen sollen) feststehen und im FSG abgebildet sind. Darauf basierend geht es darum, die vorhandenen Freiheitsgrade der fachlichen Analyse für eine Aufwandsminimierung zu nutzen.
Entscheidungsrelevant ist hier zum einen der *Aufwand für die Realisierung eines Service*. Mit zunehmendem Umfang eines Service und der dadurch höheren Komplexität der Realisierung steigt dieser Aufwand i. d. R. überproportional an, bspw. aufgrund von Seiteneffekten und Änderungen am Quellcode anderer Artefakte. Auch das Testen wird bei zunehmendem Serviceumfang aufwändiger. Zum anderen ist der *Aufwand für die Servicekomposition* mittels Sprachen wie WS-BPEL in die Entscheidung einzubeziehen. Hier fließt bspw. der Aufwand für das Suchen und Einbinden einzelner Services, der Erstellungsaufwand einer WS-BPEL-Datei oder die Vorbereitung des späteren Betriebs der Komposition ein. Schließlich gilt es auch denjenigen *Wartungs- und Pflegeaufwand* zu berücksichtigen, der durch die Wahl der Servicegranularität beeinflusst wird, bspw. falls eine Funktionalität in mehreren Services redundant realisiert wird und somit mehrfacher Wartungs- und Pflegeaufwand anfällt.
Nicht relevant für die hier betrachtete Entscheidung ist dagegen der *Einmalaufwand* für die Einrichtung einer serviceorientierten Infrastruktur, da dieser Aufwand unabhängig von der Wahl der Servicegranularität ist: Hierunter fällt das Einarbeiten in Servicestandards, das Auf-

setzen und Installieren der Infrastruktur (z. B. Engine, Verzeichnisdienst-Server, Enterprise Service Bus) sowie die Installation und das Einarbeiten in die Entwicklungsumgebung u. ä.

Zielfunktion

Das Entscheidungsproblem kann demnach wie folgt beschrieben werden: Es soll eine zulässige Lösung für die Zuordnung von Services zu Funktionalitäten (repräsentiert durch die Matrix I_{SM}) gefunden werden (zum Kriterium der Zulässigkeit s. u.), welche die gesamten Aufwände für die Realisierung C_R, die Komposition C_K und für Wartung und Pflege C_P minimiert:

$$Z(I_{SM}) = C_R(I_{SM}) + C_K(I_{SM}) + C_P(I_{SM}) \rightarrow \min!$$

Im Folgenden werden die einzelnen Aufwandsgrößen beschrieben.

Aufwand für die Servicerealisierung

Ausgangspunkt für die Schätzung des Realisierungsaufwands eines Service ist - wie oben dargestellt - primär der Umfang der zu implementierenden Funktionalität. Folgende Annahme wird getroffen:

(A3) Für einen Service s_i entsteht ein Realisierungsaufwand $c_R(s_i)$, der vom Umfang der zu realisierenden Funktionalität $size_{m_j}$ - gemessen z. B. in LOC - abhängig ist. Dabei wird angenommen, dass bspw. pro LOC ein Aufwandsatz c_{var} notwendig ist und der Aufwand mit steigendem Umfang (aus Komplexitätsgründen) überproportional zunimmt (Exponent $b > 1$). Daneben kann zur Servicerealisierung auch ein vom Umfang unabhängiger Aufwand c_{fix} auftreten (z. B. Deployment-Aufwand, wie Bekanntmachung des Service im Dienstverzeichnis).

Die Parameter c_{var} und b können dabei bspw. als Linear- und Skalierungsfaktoren ähnlich zum COCOMO-Ansatz interpretiert werden. Basierend auf (A3) ergibt sich der Realisierungsaufwand eines Service s_i zu:

$$c_R(s_i) = \sum_{j=1}^{|M|} I_{s_i,m_j} \cdot (c_{fix} + c_{\text{var}} \cdot (size_{m_j})^b) \quad \text{mit}: c_{\text{var}} > 0,\ c_{fix} \geq 0,\ b > 1$$

Zudem ist darauf hinzuweisen, dass für unterschiedliche Servicetypen bspw. die Werte der Parameter c_{var} und b auch variiert werden können. Aus Darstellungsgründen haben wir jedoch beim obigen Term für den Realisierungsaufwand $c_R(s_i)$ auf eine dafür notwendige, weitere Indexierung der Parameter verzichtet. Das später vorgestellte prototypische Softwarewerkzeug erlaubt eine solche Definition verschiedener Parameter.

Der gesamte Realisierungsaufwand einer Servicelandschaft ergibt sich somit als Summe über den Realisierungsaufwand aller Services:

$$C_R(I_{SM}) = \sum_{i=1}^{|S|} c_R(s_i)$$

Aufwand für die Servicekomposition

Wird ein Prozess mit vielen feingranularen Services anstatt mit wenigen grobgranularen Services realisiert (Granularitätsverständnis nach dem Größenmaß), so steigt der Kompositionsaufwand. Ausgangspunkt für seine Ermittlung - hier auf Ebene der Prozesse dargestellt - ist der bereits definierte Umfang der Kompositionslogik:

(A4) Der Aufwand c_K für die Servicekomposition eines Prozesses ist vom Umfang der Kompositionslogik abhängig, die nicht durch einen Service bereits realisiert ist (wiederum bei einem Aufwandsatz c_{var}^{comp}). Dabei wird angenommen, dass der Aufwand mit steigendem Umfang der Kompositionslogik (höhere Komplexität) überproportional zunimmt (Exponent $f > 1$).

Die Annahme (A4) lässt sich leicht mit Hilfe der Abbildung 1 verdeutlichen: Zur Realisierung des Prozesses m_0 sind die Services s_1, s_2, s_3 und s_4 zu komponieren. Zudem ist bei der Kompositionslogik die Funktionalität m_4 und damit $size_{m_4}^{comp}$ (da die Kompositionslogik von m_4 nicht von einem Service realisiert wird) sowie $size_{m_0}^{comp}$ für den Prozess m_0 selbst zu berücksichtigen. Der Kompositionsaufwand für einen Prozess $m_p \in P$ ergibt sich damit allgemein zu:

$$c_K(m_p) = c_{\text{var}}^{comp} \cdot \left(size_{m_p}^{comp} + comp(I_{SM}, m_p)\right)^f \quad \text{mit}: c_{\text{var}}^{comp} > 0, \quad f > 1$$

Dabei können die Werte der Parameter c_{var}^{comp} und f aufgrund der unterschiedlichen Verfahren und Sprachen für die Entwicklung von Services bzw. für deren Komposition von den Werten der Parameter c_{var} und b abweichen. Mit Hilfe von $comp(I_{SM}, m_p)$ wird anhand der Wege von Prozess m_p zu den Basisfunktionalitäten diejenigen vorausgehenden Funktionalitäten ermittelt, die nicht bereits durch einen Service realisiert wurden (wie m_4 im Beispiel). Der gesamte Kompositionsaufwand ergibt sich wiederum als Summe über den Kompositionsaufwand der einzelnen Prozesse.

Wartungs- und Pflegeaufwand durch Mehrfachrealisierung

Daneben ist auch derjenige Wartungs- und Pflegeaufwand entscheidungsrelevant, der explizit durch die Wahl der Servicegranularität entsteht bzw. vermieden werden kann. Wird eine mehrfach benötigte Funktionalität auch mehrfach realisiert, ist bei einer Änderung der Funktionalität jede Servicerealisierung anzupassen. Selbst wenn dies der Grundidee einer SOA eigentlich widerspricht, wäre eine Vernachlässigung dieses Falls unrealistisch. Insofern wird vorgeschlagen, den zusätzlichen Wartungs- und Pflegeaufwand von der Anzahl und dem Umfang der *mehrfachen* Realisierung einer Funktionalität abhängig zu machen. Grund hierfür ist, dass mit zunehmender Anzahl und zunehmendem Umfang einer anzupassenden Funktionalität auch mit steigendem Wartungs- und Pflegeaufwand (wegen Komplexität) zu rechnen ist (vgl. Keller 2007):

(A5) Durch jede mehrfache, redundante Realisierung einer Funktionalität entsteht zusätzlicher Wartungs- und Pflegeaufwand, der mit steigendem Umfang dieser Funktionalität überproportional zunimmt. Analog zu oben, wird hier ein Satz $pen_{var} > 0$ für den variablen Aufwand und ein Exponent $h > 1$ verwendet.

Basierend auf (A5) wird ermittelt, wie oft eine Funktionalität m_i direkt und indirekt mittels Services realisiert ist. In Abbildung 1 ist die Anzahl der Realisierungen der Funktionalität m_9 mit $r_{m_9} = 2$ anzugeben. Der Wartungs- und Pflegeaufwand c_p für eine Funktionalität m_i ergibt sich dann zu:

$$c_p(m_i) = \max[(r_{m_i} - 1), 0] * pen_{\text{var}} \cdot (size_{m_i})^h \quad mit: pen_{\text{var}} > 0, h > 1$$

Die *max*-Funktion stellt hier sicher, dass nur mehrfach realisierte Funktionalitäten einbezogen werden. Der gesamte Wartungs- und Pflegeaufwand C_P durch Mehrfachrealisierung ist wieder über die Summe der Aufwände $c_p(m_i)$ aller Funktionalitäten zu berechnen.

Wahl der Servicegranularität unter ökonomischen Aspekten

Anhand des FSG und der obigen Zielfunktion lässt sich nunmehr die Servicegranularität unter ökonomischen Aspekten optimieren. Dabei sind die Grundzusammenhänge berücksichtigt: Je feingranularer ein Service (bspw. nach dem Größenmaß), desto eher kann dieser mehrfach verwendet werden, ohne dass die durch den Service realisierten Funktionalitäten mehrfach implementiert werden müssen. Dies senkt ceteris paribus den Realisierungsaufwand. Abbildung 1 zeigt ein Beispiel: Die Funktionalität m_9 wird in zwei Prozessen benötigt, ist aber nicht direkt realisiert. Der Umfang dieser Funktionalität geht somit in die Berechnung von zwei vorausgehenden realisierten Funktionalitäten ein (hier m_6 und m_{11}). Der Realisierungsaufwand fällt also zweimal (für die Services s_3 und s_5) an, wohingegen bei einer direkten Realisierung der Aufwand für m_9 nur einmal anfallen würde. Darüber hinaus geht eine Mehrfachrealisierung auch mit einem erhöhten Wartungs- und Pflegeaufwand einher. Andererseits entsteht bei feingranularen Services höherer Kompositionsaufwand.

Die Ermittlung der optimalen Lösung ist nicht trivial. Es gilt die Adjazenzmatrix I_{SM} derart zu besetzen, dass die Lösung erstens zulässig ist und zweitens den Gesamtaufwand laut obiger Zielfunktion minimiert. Um die Lösung mit dem minimalen Gesamtaufwand zu ermitteln, ist ein Verfahren realisiert, das beginnend mit den Basisfunktionalitäten schrittweise diesen Services zuordnet. Danach werden sukzessive nicht mehr die Basisfunktionalitäten mit Services realisiert, sondern den vorausgehenden Funktionalitäten jeweils Services zugeordnet. Für die jeweiligen Lösungen ist zugleich immer die Zulässigkeit zu prüfen. Eine Zuordnung von Services zu Funktionalitäten im FSG ist genau dann zulässig, wenn es möglich ist, alle Prozesse mit den zugeordneten Services zu komponieren und damit zu realisieren. D. h. konkret, dass jeder mögliche Gesamtweg $w(m_0, m_n)$ im FSG mindestens eine direkt realisierte Funktionalität enthalten muss. Dies entspricht dem Zulässigkeitskriterium und ist hinrei-

chend, da eine vorausgehende Funktionalität direkt oder indirekt immer auch die über die Disaggregationsbeziehungen referenzierten Basisfunktionalitäten (Adjazenzmatrix I_{MM}) umschließt. Ist eine Lösung nicht zulässig, ist mindestens eine Basisfunktionalität $m_n \in B$ eines Gesamtweges für den Prozess $m_p \in P$ nicht verfügbar. Damit wäre der Prozess m_p nicht ausführbar. Diese Bedingungen für die Zulässigkeit einer Lösung wurden formal definiert (siehe Anhang 1) und im Werkzeug implementiert. Abschließend wird aus allen zulässigen Lösungen diejenige selektiert, deren Realisierung zum minimalen Gesamtaufwand führt.

Neben der Berechnung der Lösung sind auch die Konfigurationsgrößen für die Aufwandsfunktionen zu ermitteln. Hier kann eine Orientierung an bekannten Schätzverfahren aus der Softwareentwicklung erfolgen. Bspw. wird im COCOMO-Ansatz der Aufwand anhand der Formel $PM = A \cdot EM \cdot size^{B+SF}$ geschätzt, wobei *A* und *EM* Linearfaktoren und *B* und *SF* Skalierungsfaktoren sind. Ihre Werte sind projekt- sowie unternehmensspezifisch zu bestimmen. Bereits von Tansey und Stroulia (2007) wird COCOMO auch bei Servicerealisierungen verwendet: Sie verdeutlichen die grundsätzliche Anwendbarkeit, verweisen jedoch auch darauf, dass für die Bestimmung einzelner Faktoren oftmals noch nicht auf eine umfangreiche (unternehmensübergreifende) Datenbasis zurückgegriffen werden kann. Hier ist man - wie auch das nachfolgende Fallbeispiel zeigt - auf unternehmensinterne Schätzungen angewiesen. Dennoch sind Grundannahmen, wie bspw. der konvexe Verlauf der Aufwandsfunktionen sinnvoll. Um die Folgen einer Unschärfe der Schätzungen zu analysieren, wurde im entwickelten Softwarewerkzeug eine Sensitivitätsanalyse integriert, welche die Analyse der Robustheit der Ergebnisse unterstützt.

4 Prototypische Umsetzung und Fallbeispiel

Im Folgenden ist die Umsetzung des Entscheidungsmodells in einem Softwarewerkzeug dargestellt. Danach wird das Fallbeispiel einer Bank diskutiert.

4.1 Prototypische Realisierung

Das vorgestellte Modell wurde in Form eines Plug-in für das Open-Source-Framework Eclipse umgesetzt. Die Implementierung erfolgte in Java und basiert auf dem Eclipse Modeling Framework. Die grafischen Ein- und Ausgaben wurden mittels des Graphical Editing Framework realisiert. Für die Erstellung des FSG steht ein grafischer Editor zur Verfügung, wobei die Funktionalitäten aus einer seitlichen Palette (vgl. rechter Teil in Abbildung 3) in den Arbeitsbereich gezogen und deren Parameter (z. B. Bezeichner und Umfang der Funktionalität) eingegeben werden. Zudem sind die Funktionalitäten mit gerichteten Kanten im azyklischen FSG einzubinden. Eine weitere Maske ermöglicht nicht nur die Eingabe der Parameter der Aufwandsfunktionen. Vielmehr können diese Funktionen bspw. auch unternehmensindividuell verändert werden (um bspw. Function Points anstatt LOC zu verwenden).

Auf Basis dieser Daten wird im nächsten Schritt die aufwandsminimale Servicerealisierung berechnet. Die Ausgabe der Ergebnisse erfolgt dabei zweigeteilt: Zum einen in Form einer Tabelle, in der dargestellt ist, welche Funktionalitäten als Service zu realisieren sind, dem zugehörigen Aufwand der Lösung und den Werten der drei Granularitätsmetriken. Zum anderen wird die aufwandsminimale Lösung auch grafisch dargestellt (vgl. Abbildung 3). Hier wird jede zu realisierende Funktionalität laut Lösung mit einem Service (symbolisiert durch einen Kreis) verbunden. Insbesondere für einen schnellen Vergleich alternativer Lösungen bietet sich diese Art der Ausgabe an. Wie diskutiert, kann die Schätzung der Parameter mit einer Unschärfe behaftet sein. Deshalb verfügt das Werkzeug zusätzlich über eine Sensitivitätsanalyse. Hier stehen für jeden Parameter Eingabefelder zur Verfügung, um einen Schwankungsbereich (untere und obere Grenze bspw. für den Aufwandsatz c_{var}) zu spezifizieren. Mit diesen Eingaben wird analysiert, ob sich eine ermittelte, aufwandsminimale Lösung unter Berücksichtigung des Schwankungsbereichs verändert. Daneben ist es im Rahmen der Sensitivitätsanalyse auch möglich, ausgehend von einem einzelnen Parameterwert diesen schrittweise (automatisch) so lange verändern zu lassen, bis eine neue aufwandsminimale Lösung gefunden wird. Auch dies gibt Einblicke in die Robustheit der ermittelten Lösung, was sich gerade in der praktischen Anwendung als sehr wertvoll erwies.

4.2 Fallbeispiel

Bei Finanzdienstleistern wird derzeit die Standardisierung von Prozessen und IT Anwendungen u. a. mit dem Ziel einer Aufwandssenkung vorangetrieben. Hier gewinnt auch die Ablösung monolithischer Altsysteme durch SOA zunehmend an Bedeutung (zu weiteren SOA-Zielsetzungen vgl. Baskerville et al. 2010). Vor diesem Hintergrund wurde im Kreditbereich einer großen deutschen Bank eine SOA eingeführt. Im ersten Schritt erfolgte dazu eine fachliche Analyse, d. h. es wurden ausgehend von den Prozessen Funktionalitäten identifiziert, die in Services realisiert werden konnten. Diese Analyse ließ jedoch noch wesentliche Freiheitsgrade im Sinne alternativer Servicerealisierungen offen. Um hier nicht intuitiv bzw. orientiert an Daumenregeln vorzugehen, waren ökonomische Aspekte einzubeziehen. Beispielhaft wird der Teil eines Kreditprozesses (Ausschnitt aus dem Kreditvergabeprozess „*Vergabe Privatkredit über Internet*“) herausgegriffen, der in Abbildung 2 vereinfacht und aus Vertraulichkeitsgründen anonymisiert dargestellt ist.

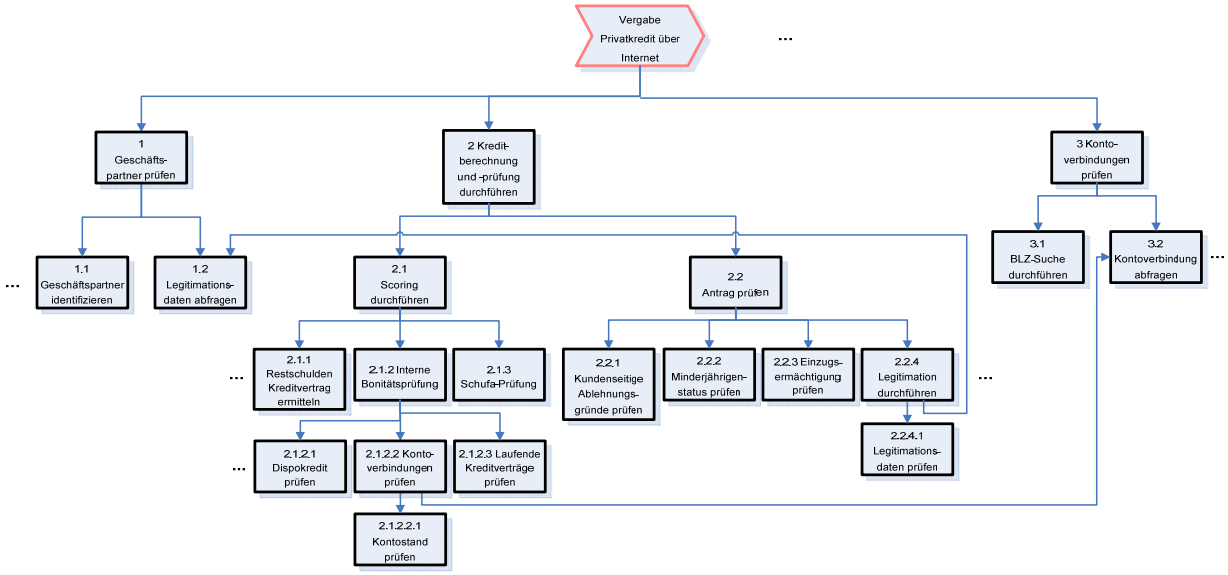


Abbildung 2: *Teil des FSG im Fallbeispiel*

Der Prozess „*Vergabe Privatkredit über Internet*“ wird hier in drei Funktionalitäten disaggregiert: „*Geschäftspartner prüfen*“ beinhaltet Funktionalitäten zur Identifikation der Partner bzw. deren Legitimation. Unter „*Kontoverbindungen prüfen*“ befinden sich Funktionalitäten wie die BLZ-Suche oder die Abfrage von Kontoverbindungen. Die dritte Funktionalität „*Kreditberechnung und -prüfung durchführen*“ wird in „*Scoring durchführen*“ und „*Antrag prüfen*“ disaggregiert. Wie in Abbildung 2 ersichtlich, werden einzelne Funktionalitäten wie bspw. „*Kontoverbindung abfragen*“ und „*Legitimationsdaten abfragen*“ mehrfach verwendet. Um den Umfang der Funktionalitäten in LOC zu bestimmen, wurde bei bereits implementierten Teilen auf den Quelltext bzw. auf vorhandene Dokumentation zurückgegriffen. Der Umfang von nicht oder nicht ausreichend dokumentierten Funktionalitäten wurde geschätzt. Hier wurde auf interne Erfahrungswerte der Aufwandsschätzung (z. B. auf Basis der Komplexität von Funktionalitäten, der verarbeiteten Daten oder hinsichtlich der verwendeten Entwicklungsverfahren) aus früheren Projekten aufgebaut. Letztlich resultierten daraus für die obigen Funktionalitäten die in Tabelle 3 dargestellten LOC. Hier umfassen die LOC für die vorausgehenden Funktionalitäten nur die Kompositionslogik und stellen nicht bereits aggregierte Werte dar (die Funktionalitäten werden im Folgenden mit der in Abbildung 2 eingeführten Nummerierung referenziert):

Tabelle 3: *Umfang der Funktionalitäten im Fallbeispiel*

Nr. der Funktionalität	*LOC*	Nr. der Funktionalität	*LOC*	Nr. der Funktionalität	*LOC*
1	200	2.1.2	600	2.2.2	2.000
1.1	1.500	2.1.3	1.500	2.2.3	800
1.2	1.000	2.1.2.1	1.600	2.2.4	200
2	400	2.1.2.2	200	2.2.4.1	1.700

2.1	1.000	2.1.2.3	1.400	3	400
2.2	450	2.1.2.2.1	400	3.1	1.000
2.1.1	1.400	2.2.1	1.000	3.2	1.400

Daneben waren auch die Konfigurationsgrößen für die Aufwandsfunktionen zu schätzen. Hierbei erfolgte eine *grundsätzliche Orientierung* an den Parametern, die in der COCOMO-Aufwandsschätzung der Bank Verwendung finden (dies kann aus mehrerlei Gründen i. A. jedoch auch problematisch sein; für den speziellen Fall der Bank erschien dies durchaus als sinnvoll). Nach eingehender Diskussion der Übertragbarkeit wurden allerdings keine Einzelwerte für diese Größen definiert. Vielmehr wurden Schwankungsbreiten festgelegt, um die inhärente Unsicherheit einer Schätzung und Übertragung zu berücksichtigen. Tabelle 4 zeigt die explizierten Bereiche:

Tabelle 4: *Schätzung der Konfigurationsgrößen für die Aufwandsfunktionen im Fallbeispiel*

Größen für den Aufwand zur Servicerealisierung	Bereiche	Größen für den Aufwand zur Servicekomposition	Bereiche
c_{var}	[2,75 - 3,25]	c_{var}^{comp}	[2,75 - 3,25]
b	[1,05 - 1,1]	F	[1,15 - 1,2]
c_{fix}	[80 - 100]		

Hintergrund für die Festlegung der Größen ist die Einstufung des Projekts seitens der Bank als mittelschweres Projekt nach COCOMO (sog. „Semi-detached mode“). Diese Einordnung erfolgte vor allem basierend auf der Analyse der Erfahrung der Projektmitarbeiter mit der Implementierung einer SOA und den zugehörigen Technologien, dem Projektumfang, der Qualität der vorhandenen Anforderungsspezifikation, der dokumentierten Schnittstellen sowie den zeitlichen Vorgaben für die Projektrealisierung. Laut COCOMO erhält ein mittelschweres Projekt einen Linearfaktor von 3,0, ein einfaches Projekt 2,4 und ein komplexes Projekt 3,6. Der Skalierungsfaktor ist bei einem mittelschweren Projekt 1,12, bei einem einfachen Projekt 1,05 und bei einem komplexen Projekt 1,2. Daran orientiert wurden für die Aufwandssätze c_{var} und c_{var}^{comp} (Linearfaktoren) jeweils die Schwankungsbreiten mit [2,75 - 3,25] festgelegt. Zwar erschien es zunächst sinnvoll, die Werte für den Parameter c_{var}^{comp} und damit den Realisierungsaufwand für die Kompositionslogik tendenziell höher festzulegen, da im Vergleich die Realisierung der Basisfunktionalitäten (im betrachteten Projekt) weniger komplex erschien. Jedoch entschied sich der Finanzdienstleister aufgrund der Erfahrungen in früheren Projekten dafür, die Bereiche für beide Aufwandssätze identisch zu wählen. Auch bei den Exponenten *b* und *f* (Skalierungsfaktoren) erfolgte eine Orientierung an den Parametern nach COCOMO. Hier wurden die definierten Bereiche mit $b \in [1{,}05 - 1{,}1]$ bzw. $f \in [1{,}15 - 1{,}2]$ festgelegt. Durch die Definition derartiger Schwankungsbereiche kann somit analysiert werden, ob eine Servicerealisierung als Lösung identifiziert wurde, die sich bereits bei klei-

nen Schätzfehlern als wenig robust erweist. Auf eine Berücksichtigung des Wartungs- und Pflegeaufwands durch Mehrfachrealisierung wurde seitens der Bank zunächst verzichtet.

4.3 Ergebnisse

Abbildung 3 zeigt im Softwarewerkzeug für den Teilgraph in Abbildung 2 die aufwandsminimale Lösung. Die Services s_1 bis s_8 (Kreise) sind den realisierten Funktionalitäten (Rechtecke) zugeordnet.

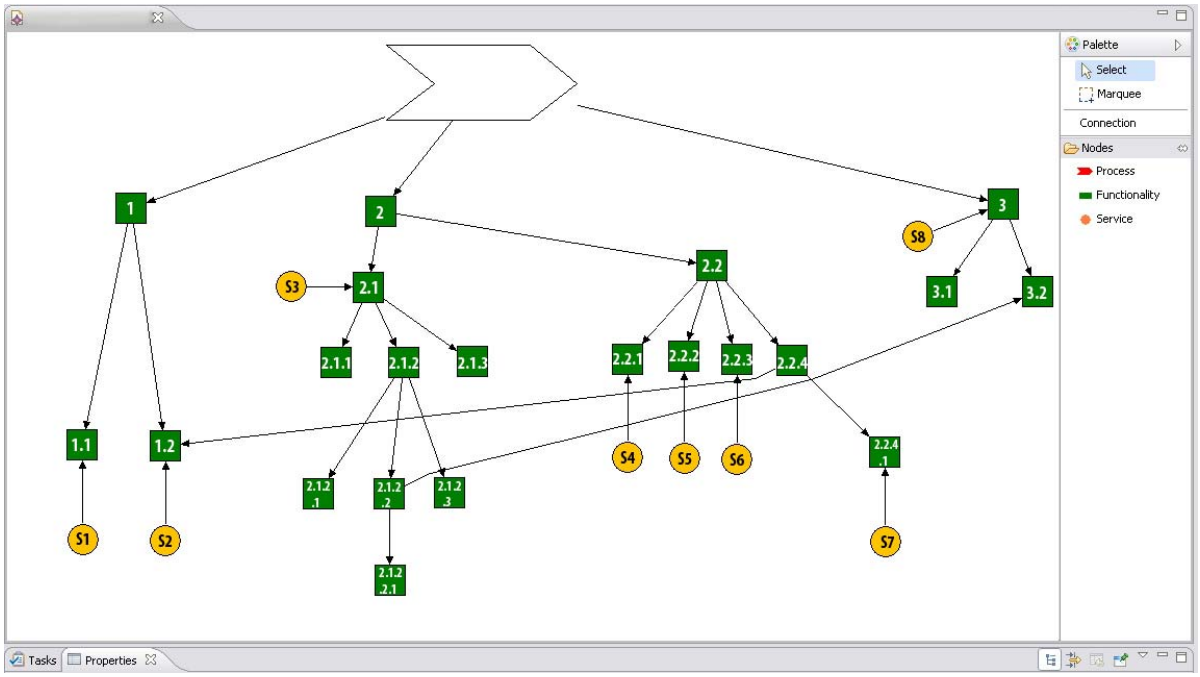


Abbildung 3: *Zuordnung der Services im Fallbeispiel*

Die obige Lösung, die einen Gesamtaufwand von insgesamt *129.493* GE aufweist, ist relativ feingranular. So besitzt das Tiefenmaß einen Wert von *0,79*, das Breiten- und Tiefenmaß einen Wert von *0,81* und das Größenmaß einen Wert von *0,67* (zur jeweiligen Berechnung dieser Metrikwerte als auch des Gesamtaufwands wird auf den Anhang 2 verwiesen). An vorausgehenden Funktionalitäten wurden „*Scoring durchführen*“ (2.1) und „*Kontoverbindungen prüfen*“ (3) mit je einem Service realisiert. Dies führt bspw. zu einer Mehrfachrealisierung der Basisfunktionalität „*Kontoverbindung abfragen*“. Durch den relativ geringen Umfang der realisierten Funktionalitäten ist die Mehrfachrealisierung dennoch ökonomisch vorteilhaft. Im Beispiel ist leicht zu erkennen, dass es erforderlich ist, jeweils eine individuelle Analyse des vorliegenden FSG vorzunehmen. Eine Daumenregel im Sinne einer grundsätzlich grob- oder feingranularen Realisierung hätte im vorliegenden Fall nicht zu dem ökonomisch besten Ergebnis geführt, da sich bspw. die beiden „Nachbar“-Funktionalitäten „*Scoring durchführen*“ (2.1) und „*Antrag prüfen*“ (2.2) in der Servicerealisierung und damit in der Granularität grundlegend unterscheiden. Interessant ist zudem folgender Punkt: Lässt man die Optimierung mit Hilfe des Werkzeugs mehrere tausendmal wiederholen, wobei die Werte für die Konfigurati-

onsgrößen der Aufwandsfunktionen jeweils zufällig aus den definierten Schwankungsbereichen gezogen werden, so ergibt sich folgendes Resultat: Die obige Lösung bleibt in über 85% der Durchläufe die aufwandsminimale Lösung. In den anderen 15% ist diese Lösung entweder die zweit- oder drittbeste Lösung, wobei die dann besseren Lösungen sich in allenfalls einer geänderten Servicezuordnung unterscheiden. Diese Analyse reduziert zusammen mit der Erfahrungswert-basierten Parameterschätzung im Beispiel erheblich die Unsicherheit eine wenig robuste Lösung ermittelt zu haben.

5 Zusammenfassung und Forschungsbedarf

Im Beitrag wurde ein Entscheidungsmodell vorgestellt, das die Wahl einer adäquaten Servicegranularität unter ökonomischen Aspekten unterstützt. Damit soll eine vorgelagerte fachliche Analyse (Domain Analysis) erweitert werden. Die Vielzahl möglicher Servicerealisierungen sowie die komplexen, gegenläufigen Aufwandswirkungen machen eine manuelle Optimierung sehr schwierig oder gar unmöglich. Daher wurde ein Werkzeug entwickelt und sein Einsatz anhand eines Fallbeispiels demonstriert. Weiterhin hat sich im Zuge der Anwendung gezeigt, dass die in der Praxis oft propagierten Daumenregeln zur Servicegranularität (z. B. Helbig und Scherdin 2008) durchaus kritisch zu sehen sind. So kann bspw. die Forderung nach möglichst grobgranularen Services systematisch zu ökonomisch schlechten Lösungen führen. Ursachen hierfür sind der überproportional ansteigende Realisierungsaufwand und eine ggf. erforderliche Mehrfachrealisierung von Funktionalitäten in verschiedenen Services. Für eine Entscheidung ist es vielmehr erforderlich, den zugrunde liegenden Funktionalitätsgraph zu analysieren, um dann abhängig von seiner konkreten Ausprägung die adäquate Servicegranularität zu bestimmen. So kann es durchaus ökonomisch vorteilhaft sein, die Funktionalitäten eines Prozesses unterschiedlich granular zu realisieren (vgl. auch das Fallbeispiel in Abbildung 3). Mit dem Modell und dem Werkzeug sind damit Instrumente verfügbar, um ökonomische Einflüsse abzubilden. Darüber hinaus wurden auch drei formal definierte Metriken vorgestellt und diskutiert, die eine nachvollziehbare Granularitätsmessung erlauben.

Die ökonomische Betrachtung erweitert die fachlichen Ansätze zur Serviceidentifikation. Primär ist die Kompatibilität zu fachlich orientierten Ansätzen dann gegeben, falls die Grundstruktur des Funktionalitätsgraphen aus diesen Ansätzen entwickelt werden kann: Bspw. schlägt Winkler (2007) vor, Aktivitätsdiagramme schrittweise zu verfeinern und anschließend so aufzubereiten, dass mehrfach in Prozessen vorkommende Funktionalitäten zusammengefasst werden können. Das Ergebnis ist ein unserem Funktionalitätsgraph entsprechender gerichteter Graph. Die Entscheidung, welche Funktionalitäten in einem Service realisiert werden sollen, trifft Winkler rein argumentativ auf der Grundlage verschiedener Annahmen (bspw. möglichst hohe Mehrfachverwendung). Hier kann unser Modell ökonomische Zu-

sammenhänge explizieren und eine zusätzliche Entscheidungshilfe bieten. Auch andere Ansätze nutzen als Grundlage für die Serviceidentifikation eine Analyse der Funktionen einer Domäne bzw. eine Funktionsdekomposition (bspw. Fiege 2009, Offermann 2008), wobei je nach Zerlegung Servicekandidaten unterschiedlicher Granularität entstehen können. Entsprechend kann auch dort - nach einer Aufbereitung – das vorgestellte Entscheidungsmodell ergänzend zum Einsatz kommen.

Daneben sind auch kritische Punkte anzumerken, die zugleich den weiteren Forschungsbedarf definieren: Erstens ist für die Anwendung des Entscheidungsmodells die Erstellung eines Funktionalitätsgraphs nötig, der wiederum Ergebnis - wie im Fallbeispiel des Finanzdienstleisters - einer fachlichen Analyse ist. Hier stellt sich jedoch die Frage, ob und wie ggf. mit einer unvollständigen Datenbasis (bspw. nur ein Teil des Funktionalitätsgraphs ist modelliert) eine robuste Lösung im Sinne der Servicegranularität ermittelbar ist. Letzteres bedeutet, dass sich die ermittelte Lösung bei einer Anpassung oder Komplettierung der Datenbasis nicht mehr grundlegend ändern sollte. Dabei ist allerdings auch klar, dass für eine fundierte Entscheidung eine gewisse Datenbasis verfügbar sein muss. Analog ist bei den Konfigurationsgrößen für die Aufwandsfunktionen zu verfahren. Auch hier ist man darauf angewiesen, dass qualitätsgesicherte Schätzungen erfolgen. Jedoch sind diese auch ohne Einsatz eines Entscheidungsmodells notwendig, will man eine seriöse Schätzung des Realisierungsaufwands im Projekt machen. Forschungsbedarf besteht hier insofern, die Verfahren zur Bestimmung der Linear- und Skalierungsfaktoren gerade für die Serviceentwicklung zu adaptieren und auch die Sensitivitätsanalyse - welche die Unschärfe der Schätzungen fokussiert - auszubauen. Um das Modell auch bei unvollständiger Datenbasis anwenden zu können, sind ferner zusätzliche Erweiterungen denkbar. Bspw. könnten bestehende Schätzverfahren angepasst und in das Werkzeug integriert oder Standardwerte für Konfigurationsgrößen hinterlegt werden. Daneben sind auch etwaige Änderungen des FSG zu berücksichtigen. Hier ist es notwendig, bspw. für Prozessänderungen Wahrscheinlichkeiten (Szenarien) im Graph zu hinterlegen. Damit können zukünftige Anpassungen bei den Funktionalitäten strukturiert dargestellt und für die zugrunde gelegten Szenarien bspw. Erwartungswerte für den Aufwand ermittelt werden. Diese können dann wiederum in die Wahl der Servicegranularität eingehen. Der entwickelte Ansatz bietet hierfür jeweils eine geeignete Ausgangsbasis.

Literatur

Aier, S. (2006): How Clustering Enterprise Architectures helps to Design Service-Oriented Architectures. In: IEEE International Conferences on Services Computing, Chicago, Illinois, USA, S. 269-272.

Albani, A.; Overhage, S.; Birkmeier, D. (2008): Towards a Systematic Method for Identifying Business Components. In: CBSE 2008, Springer, LNCS 5282, S. 262-277.

Arsanjani, A.; Ghosh, S.; Allam, A.; Abdollah, T.; Ganapathy, S.; Holley, K. (2008): SOMA: A method for developing service-oriented solutions. In: IBM Systems Journal, Vol. 47 (3), S. 377-396.

Baskerville, R. L.; Cavallari, M.; Hjort-Madsen, K.; Pries-Heje, J.; Sorrentino, M.; Virili, F. (2010): The strategic value of SOA: a comparative case study in the banking sector. In: International Journal of Information Technology and Management, 9 (1), S. 30-53.

Beverungen, D.; Knackstedt, R.; Müller, M. (2008): Entwicklung Serviceorientierter Architekturen zur Integration von Produktion und Dienstleistung – Eine Konzeptionsmethode und ihre Anwendung am Beispiel des Recyclings elektronischer Geräte. In: Wirtschaftsinformatik, 50 (3), S. 220-234.

Biffl, S.; Aurum, A.; Böhm, B.; Erdogmus, H.; Grünbacher, P. (2005): Value-Based Software Engineering. Springer, Berlin.

Birkmeier, D.; Klöckner, S.; Overhage, S. (2008): Zur systematischen Identifikation von Services: Kriterien, aktuelle Ansätze und Klassifikation. In: Modellierung betrieblicher Informationssysteme (MobIS 2008), LNI, P-141, Gesellschaft für Informatik, Bonn, S. 255-272.

Birkmeier, D.; Overhage, S. (2009): On Component Identification Approaches – Classification, State of the Art, and Comparison. In: CBSE 2009, Springer, LNCS 5582, S. 1-18.

Boehm, B. W.; Abts, C.; Brown, W.; Chulani, S.; Clark B.; Horowitz, E.; Madachy, R.; Reifer, D.; Steece, B. (2000): Software Cost Estimation with COCOMO II. Prentice Hall, Upper Saddle River, New Jersey.

Boerner, R.; Goeken, M. (2009): Identification of Business Services: Literature review and lessons learned. In: Proceedings of the 15th Americas Conference on Information Systems (AMCIS 2009), San Francisco, USA.

Buhl, H. U.; Heinrich, B.; Henneberger, M.; Krammer, A. (2008): Service Science. In: Wirtschaftsinformatik 50 (1), S. 60-65.

Erl, Thomas (2005): Service-Oriented Architecture – Concepts, Technology, and Design. Prentice Hall, Upper Saddle River, New Jersey.

Erl, T. (2007): SOA Principles of Service Design. Prentice Hall, Upper Saddle River, New Jersey.

Fiege, R. (2009): Axiomatic Design: Eine Methode zur serviceorientierten Modellierung. Gabler, Wiesbaden.

Galster, M., Bucherer, E. (2008): A Business-Goal-Service-Capability Graph for the Alignment of Requirements and Services. In Proceedings of the IEEE Congress on Services, S. 399-406.

Haesen, R.; Snoeck, M.; Lemahieu, W.; Poelmans S. (2008): On the Definition of Service Granularity and Its Architectural Impact, Lecture Notes In Computer Science; Vol. 5074 archive, Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAISE), S. 375-389.

Helbig, J., Scherdin, A. (2008): Wie granular müssen SOA-Dienste sein? In: Computerwoche 5/2008, S. 14.

Henning, H.-J. (2007): Granularität von Services – Kriterien zur Strukturierung einer Servicelandschaft. In: Gernot Starke, Stefan Tilkov (Hrsg.): SOA-Expertenwissen. dpunkt, Heidelberg, S. 343-356.

Her, J.S.; La, H. J.; Kim, S.D. (2008): A Formal Approach to Devising a Practical Method for Modeling Reusable Services. In: IEEE International Conference on e-Business Engineering (ICEBE), Xi'an, China, October, S. 221-228.

IBM (1984): Business Systems Planning – Information Systems Planning Guide, International Business Machines, Atlanta.

Joachim, N.; Beimborn, D.; Weitzel, T. (2011): SOA-Governance für effektive serviceorientierte Architekturen – Eine empirische Studie in der deutschen Dienstleistungswirtschaft. In: Bernstein, A., Schwabe, G. (Hrsg.): Tagungsband der 10. Internationalen Tagung Wirtschaftsinformatik, Band 1, Zürich, Schweiz, S. 446-455.

Keller, W. (2007): Regeln und Heuristiken für gutes Schnittstellendesign. In: Starke, G.; Tilkov; S. (Hrsg.): „SOA-Expertenwissen". dpunkt, Heidelberg, S. 357-380.

Kim, S. D.; Chang, S. H. (2004): A Systematic Method to Identify Software Components. In: Proceedings of the 11th Asia-Pacific Software Engineering Conference, S. 538-545, Busan, Korea.

Krafzig, D.; Banke, K.; Slama, D. (2005): Enterprise SOA – Service-Oriented Architecture Best Practices. Prentice Hall, Upper Saddle River, New Jersey.

Lee, J.K.; Jung, S.J.; Kim, S.D.; Jang, W.H.; Ham, D.H. (2001): Component Identification Method with Coupling and Cohesion, Proc. of Asia-Pacific Software Engineering Conference (APSEC'01); Macau, China, S. 79-87.

Melzer, I.; Dostal, W.; Jeckle, M.; Zengler, B. (2007): Service-orientierte Architekturen mit Web Services: Konzepte - Standards - Praxis. Spektrum Akademischer Verlag, Heidelberg.

Offermann, P. (2008): SOAM – Eine Methode zur Konzeption betrieblicher Software mit einer Serviceorientierten Architektur. In: Wirtschaftsinformatik 2008 (6), S. 461-471.

Papazoglou, M. P. (2003): Service-Oriented Computing: Concepts, Characteristics and Directions. In: Fourth International Conference on Web Information Systems Engineering (WISE'03), S. 3-12.

Papazoglou, M. P.; Traverso, P.; Dustdar, S.; Leymann, F.; Krämer, B. J. (2006): Service Oriented Computing Research Roadmap. In: Dagstuhl Seminar Proceedings 05462, Service Oriented Computing (SOC), online verfügbar unter http://drops.dagstuhl.de/opus/volltexte/2006/524, S. 1-29.

Papazoglou, M. P.; van den Heuvel, W.-J. (2006) Service-Oriented Design and Development Methodology. In: International Journal of Web Engineering and Technology 2 (4), S. 412-442.

Richter, J.-P.; Haller, H.; Schrey, P. (2005): Serviceorientierte Architektur. In: Informatik-Spektrum 28 (5), S. 413-416.

Schelp, J.; Winter, R. (2008): Entwurf von Anwendungssystemen und Entwurf von Enterprise Services – Ähnlichkeiten und Unterschiede. In: Wirtschaftsinformatik 50 (1), S. 6-15.

Tansey, B.; Stroulia, E. (2007): Valuating Software Service Development: Integrating COCOMO II and Real Options Theory. In: ICSE2007 Workshops, Minneapolis, USA, S. 8, May.

Thomas, O.; Leyking, K.; Scheid, M. (2010): Serviceorientierte Vorgehensmodelle: Überblick, Klassifikation und Vergleich. In: Informatik-Spektrum, 33 (4), S. 363-379.

Wang, Z.-J.; Xu, X.; Zhan, D. (2005): A Survey of Business Component Identification Methods and Related Techniques. In: International Journal of Information Technology 2 (4), S. 229-238.

Wang, Z.-J.; Zhan D.-C.; Xu, X.-F. (2006): STCIM: A Dynamic Granularity Oriented and Stability Based Component Identification Method. In: ACM SIGSOFT Software Engineering Notes 31 (3), S. 1-14.

Wehrmann, A.; Gull, D. (2006): Ein COCOMO-basierter Ansatz zur Entscheidungsunterstützung beim Offshoring von Softwareentwicklungsprojekten. In: Wirtschaftsinformatik 48 (6), S. 407-417.

Winkler, V. (2007): Identifikation und Gestaltung von Services – Vorgehen und beispielhafte Anwendung im Finanzdienstleistungsbereich. In: Wirtschaftsinformatik 49 (4), S. 257-266.

Winter, R. (2003): An Architecture Model for Supporting Application Integration Decisions. In: Ciborra, C. U.; Mercurio, R.; de Marco, M.; Martinez, M.; Carignani, A. (Hrsg.): Proceedings of the 11th European Conference on Information Systems (ECIS 2003), Naples, Italy.

Anhang 1: Prüfung der Zulässigkeit einer Lösung

Folgende Bedingungen für die Zulässigkeit einer Lösung sind abzuprüfen:

1) Jede Funktionalität muss mindestens einer übergeordneten Funktionalität oder einem Prozess zugeordnet sein (nur einmal für den gesamten Graph zu prüfen!)

$$\sum_i I_{m_i, mj} + \sum_i I_{p_i, mj} \geq 1 \quad \forall \; j$$

2) Für jede aufgrund des kombinatorischen Zuordnungsproblems ermittelte Lösung muss gelten: Selektiere einen beliebigen Prozess *p'.* Prüfe nun für jeden Pfad, beginnend mit der Kante $\{p', m'\} \in E$, ob für die Funktionalität *m'* gilt: $\{s, m'\} \in E$.
   a) Falls dies der Fall ist, so gehe zum nächsten Pfad, d. h. der Kante $\{p', m''\} \in E$ usw. Falls es keine entsprechende Kante mehr gibt, dann führe die beschriebene Prozedur für jeden anderen Prozess *p''* ebenfalls durch.
   b) Falls dies nicht der Fall ist, so prüfe für jeden Pfad, beginnend mit Kante $\{m', m'''\} \in E$, ob für die Funktionalität *m'''* gilt: $\{s, m'''\} \in E$. Falls dies der Fall ist, so prüfe den nächsten Pfad mit der Kante $\{m', m''''\} \in E$. Ist dies nicht der Fall, dann prüfe alle Pfade mit der Kante $\{m''', m''''\} \in E$ usw.
      ⇨ Falls für einen Pfad $\{p', m'\}, \{m', m''\}, \ldots, \{m''', m''''\} \in E$ sowie *m''''* als elementare Funktionalität gilt, dass $\{s, m'\}, \{s, m''\}, \ldots, \{s, m'''\}, \{s, m''''\} \notin E$, dann ist die Lösung keine zulässige Lösung.

Für jeden Prozess *p'* muss gelten, dass alle die im Prozess enthaltenen Basisfunktionalitäten (über die Kanten $\{p', m\}, \{m, m\} \in E$ ermittelbar) über Services direkt oder indirekt implementiert sind.

Anhang 2: Berechnung des Gesamtaufwands und der Metrikwerte im Beispiel

Der Gesamtaufwand für die Realisierung und die Komposition lässt sich für die in Abbildung 3 dargestellte Servicerealisierung wie folgt berechnen (der Aufwand für Wartung und Pflege C_P blieb im Beispiel unberücksichtigt). Es gilt:

$$Z(I_{SM}) = C_R(I_{SM}) + C_K(I_{SM})$$

mit der Matrix I_{SM}: *S x M* =

$I_{SxM} :=$

	m_1	$m_{1.1}$	$m_{1.2}$	m_2	$m_{2.1}$	$m_{2.1.1}$	$m_{2.1.2}$	$m_{2.1.3}$	$m_{2.1.2.1}$	$m_{2.1.2.2}$	$m_{2.1.2.3}$	$m_{2.1.2.2.1}$	$m_{2.2}$	$m_{2.2.1}$	$m_{2.2.2}$	$m_{2.2.3}$	$m_{2.2.4}$	$m_{2.2.4.1}$	m_3	$m_{3.1}$	$m_{3.2}$
S_1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
S_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
S_7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
S_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

als Servicezuordnung (mit $I_{s_i,m_j} = 1$ wenn Funktionalität m_j als Service s_i realisiert wird, ansonsten gilt $I_{s_i,m_j} = 0$). Legt man die einzelnen Parameterwerte c_{var} = 3,0, c_{fix} = 90 und b = 1,075 zugrunde, ergeben sich mit dem Term $c_R(s_i) = \sum_{j=1}^{|M|} I_{s_i,m_j} \cdot (c_{fix} + c_{\text{var}} \cdot (size_{m_j})^b)$ folgende Aufwände in GE für die Services s_1 bis s_8:

Service	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
Aufwand $c_R(s_i)$	**7.877,88**	**5.126,41**	**56.736,64**	**5.126,41**	**10.700,32**	**4.052,26**	**8.999,51**	**15.324,08**

Beispielhaft errechnet sich der geschätzte Aufwand $c_R(s_1)$ für Service s_1 wie folgt:

$$c_R(s_1) = I_{s_1,m_{1.1}} \cdot (c_{fix} + c_{\text{var}} \cdot (size_{m_{1.1}})^b) = 1 \cdot (90 + 3 \cdot 1.500^{1,075}) = 7.877,88\ GE$$

Über alle Services s_1 bis s_8 ergibt sich somit ein geschätzter Realisierungsaufwand C_R von insgesamt 113.943,51 GE.

Daneben ist der Aufwand c_K für die Servicekomposition des Prozesses *„Vergabe Privatkredit über Internet“* zu ermitteln. Dieser Aufwand umschließt denjenigen Umfang der Kompositionslogik, der nicht bereits durch einen Service realisiert ist. Wie aus Abbildung 3 ersichtlich, wird die Kompositionslogik der Funktionalitäten m_1, m_2, $m_{2.2}$ und $m_{2.2.4}$ zusammen mit der Prozesskompositionslogik von 200 LOC nicht von einem Service direkt oder indirekt realisiert. Damit ergibt sich eine Kompositionslogik von insgesamt 1.450 LOC. Der geschätzte Kompositionsaufwand für den Prozess im Fallbeispiel errechnet sich mit den einzelnen Parameterwerten c_{var}^{comp} = 3,0 und f = 1,175 zu:

$$c_K(m_p) = c_{\text{var}}^{comp} \cdot \left(size_{m_p}^{comp} + comp(I_{SM}, m_p)\right)^f = 3 \cdot (1.450)^{1,175} = 15.549,94\ GE$$

Addiert man die Aufwände für die Servicerealisierung und -komposition, so weist die Lösung in der obigen Matrix I_{SM} einen minimalen Gesamtaufwand von insgesamt 129.493 GE aus.

Für den im Fallbeispiel dargestellten Funktionalitäts- und Servicegraph (Abbildung 3) lassen sich ebenfalls die Werte für die drei Metriken berechnen und interpretieren. Für die Services s_1 bis s_8 ergeben sich folgende Metrikwerte für das Tiefenmaß, Breiten- und Tiefenmaß sowie das Größenmaß:

Service	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	*Gesamt*
Metrik Tiefenmaß	**1**	**1**	**0,36**	**1**	**1**	**1**	**1**	**0**	**0,79**
Metrik Breiten- und Tiefenmaß	**1**	**1**	**0,5**	**1**	**1**	**1**	**1**	**0**	**0,81**
Metrik Größenmaß	**0,44**	**0,79**	**0,44**	**0,94**	**0,88**	**0,95**	**0,90**	**0**	**0,67**

Wie die Tabelle verdeutlicht, sind die Metrikwerte für das Tiefenmaß bzw. das Breiten- und Tiefenmaß mit 0,79 bzw. 0,81 sehr ähnlich. Dies liegt daran, dass sechs von acht Services Basisfunktionalitäten realisieren und somit sinnvollerweise eine Granularität von eins (maximal feingranular) aufweisen. Nur bei Service s_3 gibt es Abweichungen: Hier gibt der Wert 0,36 des Tiefenmaßes an, dass der Service nach ca. 1/3 aller Wege zwischen dem Prozess „*Vergabe Privatkredit über Internet*“ und dem durch Service s_3 indirekt realisierten Basisfunktionalitäten liegt. Dagegen gibt das Breiten- und Tiefenmaß einen Wert von 0,5 an. Dies sagt aus, dass der Service s_3 50% aller Funktionalitäten realisiert, die Bestandteil des Teilgraphen (mit der Kante (m_p, m_2)) sind. D. h. würde Service s_3 anstatt die Funktionalität 2.1 „*Kreditberechnung und -prüfung durchführen*“ die vorangehende Funktionalität 2 „*Scoring durchführen*“ realisieren, dann verdoppelt der Service seinen Umfang an realisierten Funktionalitäten und wäre damit maximal grobgranular.
Daneben werden auch die Metrikwerte des Größenmaßes ausgewiesen. Sie beziehen sich auf den Umfang der Funktionalitäten gemessen in LOC. Insgesamt liegt der Wert mit 0,67 etwas unterhalb der Metrikwerte der anderen beiden Maße, was daran liegt, dass mit dem Service s_1 die relativ große Basisfunktionalität „*Geschäftspartner identifizieren*“ (im Verhältnis zum Teilgraph mit der Kante (m_p, m_1)) realisiert wird.